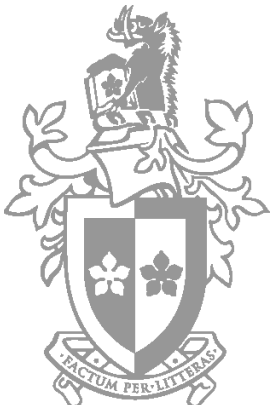


Doubtfire Helpdesk Ticketing System

SWE40001–Software Engineering Project A

Swinburne University of Technology

Team 20

Note to assessor(s): Every effort was made to convert our interactive Wiki at `http://github.com/final-year-project/portfolio-s1/wiki` in this document. This was made possible by converting the pages to LaTeX using Pandoc. Should there be formatting issues, please refer to the original content on the GitHub wiki instead.

Declaration

We declare that this portfolio is a group assessment and that no part of this submission has been copied from any other student's work or from any other source except where due acknowledgment is made explicitly in the text, nor has any part been written for us by another person.

We declare that all work in this portfolio has been sighted, reviewed and agreed by each team member, including a peer-validated review of worklog entries by each of us.

— June 1, 2016

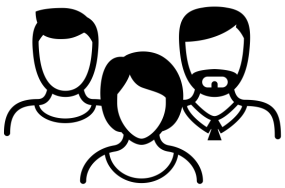
Alex Cummaudo

Jake Renzella

Lachlan West

Reuben Wilson

Contents



Chapter 1

Group Details

Contents

1 Alex Cummaudo

Student Number: 1744070
Email: acummaudo@swin.edu.au
GitHub: alexcu
Roles:

- Team Lead
- Primary Developer
- UX/UI Developer
- Documentation

2 Jake Renzella

Student Number: 9993851
Email: jrenzella@swin.edu.au
GitHub: jakerenzella
Roles:

- Testing Co-Lead
- Project Developer
- UX/UI Developer
- Documentation
- Filming & Media

3 Lachlan West

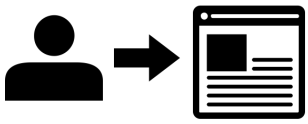
Student Number: 2423480
Email: 2423480@student.swin.edu.au
GitHub: lachlanwest
Roles:

- Project Developer
- Documentation

4 Reuben Wilson

Student Number: 9988289
Email: rwilson@swin.edu.au
GitHub: Reubsinit
Roles:

- Testing Co-Lead
- Project Developer
- UX/UI Developer
- Documentation
- Filming & Media



Chapter 2

Who Did What?

Contents

1 Alex Cummaudo

1.1 Documentation Contributions

- Wrote the Project Tools document
- Wrote the SDLC plan
- Wrote the Coding Standards Documentation for both the API and Web repos
- Wrote supporting documentation for the prototype
- Reviewed the Assessment Criteria Agreement
- Wrote most meeting minutes (refer to the revision history[1])
- Wrote use cases (Section 3) of the Technical Requirements Documentation
- Designed the Architecture Diagram in the Technical Requirements Documentation
- Reviewed the Project Plan document
- Wrote Section 6.2 of the Project Plan
- Added budget graph to Project Plan
- Worked on Presentation Storyboard

1.2 Software Contributions

Refer to my self assessment document for more detail on contributions made to software. In total I made 1,518 changes in the lines of code for the `doubtfire-web` repository (Web) and 1,694 changes in the lines of code for the `doubtfire-api` repository (API).

- API PR 1 - Add graded task functionality
- API PR 3 - Add docker as a platform option
- API PR 11 - Add new logger to subclass Rails logger
- API PR 12 - Wrap `Terminator` in a helper called `timeout_helper`
- Web PR PR 1 - Add graded tasks to feedback and admin
- Web PR 6 - Modularise all common components
- Web PR 7 - Refactor structure of error states
- Web PR 8 - Refactor structure to enhance modularity of components
- Web PR 9 - Add docker as a platform option
- Web PR 14 - Refactor all styling to use modularised SASS

[1] See `https://github.com/final-year-project/portfolio-s1/wiki/Meeting-Minutes/_history`

- Web PR 21 - Switch to BrowserSync for watch server
- Web PR 30 - Refactor portfolio wizard

1.3 Design Contributions

- Worked on the iOS application design images using Keynote
- Implemented these images into an interactive prototype using Invision

1.4 Research Contributions

- Researched the use of Socket.IO and Redis on a Rails server
- Researched parts of Minitest whilst helping Jake and Reuben

1.5 Project Management Contributions

As the project manager, I was consistently viewing our Trello Board for changes. Please refer to the Trello Board itself: `https://trello.com/doubtfire`.

- Created Trello Board for Helpdesk Ticketing System
- Created many cards related to the project and final year project deliverables
- Created workflow documentation to assist team members on how to use Trello and the process we adopted

1.6 Meeting Contributions

- Usually called most meetings, especially with our supervisor
- Made notes of meeting minutes during most meetings toward the later-half of the semester

1.7 Testing Contributions

- Helped Jake and Reuben with their exploratory testing strategy using Minitest

1.8 Presentation Contributions

- Assisted in filming stock footage of helpdesk (this also involved printing out warning signs)

- Assisted in filming general stock footage of Swinburne
- Assisted Jake with editing on parts of the presentation
- Filmed the animations needed to be created
- Filmed the prototype in use

1.9 Other Contributions

- Created the Wiki and then converted it into a LaTeX document
- Helped everyone get started with the project; teaching git and pull requests and the git workflow
- Crash course on CoffeeScript given to team members
- Helped each team member install Doubtfire on their machines

2 Jake Renzella

2.1 Documentation Contributions

- Wrote meeting minutes (refer to the revision history[2])
- Created most of the Project Plan document alongside Reuben
- Created the modified Gantt Chart for Project Plan
- Reviewed the project plan
- Worked on Presentation Storyboard
- Created App flowcharts alongside Alex

2.2 Software Contributions

- 10 Pull Requests in Doubtfire Web accepted in production detailed in learning summary
- third most contributed developer while been on project for least amount of time (as top 3)
- 2 pull requests on Doubtfire API

[2]See `https://github.com/final-year-project/portfolio-s1/wiki/Meeting-Minutes/_history`

2.3 Design Contributions

- Worked on the iOS application design images using Keynote
- Implemented these images into an interactive prototype using Invision

2.4 Research Contributions

- Research possible testing suites for Rails
- Research Minitest framework for Rails testing
- Research BLE technology for use in Ticketing Solution

2.5 Project Management Contributions

- Assist Alex and team in Trello and Slack management
- Properly utilise Trello for development purporses.

2.6 Meeting Contributions

- Made it to most to all meetings
- took miniutes at meetings

2.7 Testing Contributions

- Researched possible testing frameworks and decided with Minitest (with Andrew)
- Research Minitest application to Rails server
- Wrote test integration with Rails server
- Wrote tests
- (Note that due to Git complications, Reuben has absorbed all of my testing commits on Github, as a result they will not show)

2.8 Presentation Contributions

- Created the video plan
- filmed stock footage at Swinburne with Alex
- filmed timelapses at Swinburne with team
- maneged helpdesk when filming

- managed equipment
- wrote and filmed interviews with students, staff.
- filmed animations for video
- edited entire presentation video

3 Lachlan West

3.1 Documentation Contributions

- Contributed to SEP document
- Collaborated on Storyboarding

3.2 Software Contributions

- 1 API PR recorded in release card[3]
- 1 Web PR recorded in release card[4]
- 1 other API PR not approved but mostly complete card[5]

3.3 Design Contributions

- Contributed to Design documentation (use case, domain diagram)

3.4 Project Management Contributions

- Use of Trello and Slack to communicate and collaborate with team.

3.5 Meeting Contributions

- Attended all client and supervisor meetings that I was made aware of.
- Participated in discussing requirements and design with team and our client.

[3]See `https://github.com/doubtfire-lms/doubtfire-api/pull/13`
[4]See `https://github.com/doubtfire-lms/doubtfire-web/pull/26`
[5]See `https://github.com/doubtfire-lms/doubtfire-api/pull/19s`

3.6 Presentation Contributions

- Helped with the filming for the presentation video.
- Helped with storyboarding and overall direction of video.
- Attended all presentations, both mid semester and video presentations.

4 Reuben Wilson

4.1 Documentation Contributions

- Worked on Project Plan

4.2 Software Contributions

- Added Unit Code to units in unit drop down menu
- Remove duplicate students from enrolment page
- Renamed fix_and_include to fix_and_resubmit
- Added number of times submitted to submissions
- Extended student details on LaTex generated submission PDFS

4.3 Research Contributions

- Assist in researching Minitest framework in order to implement into DoubtFire API test

4.4 Testing Contributions

- Assist in implementing Minitest framework into DoubtFire API test cases
- Implement tests using Minitest to test authentication API endpoint

4.5 Presentation Contributions

- Assisted in filming stock footage of helpdesk
- Assisted in filming interview footage for presentation video

Chapter 3

Assessment Criteria Agreement

Contents

1 Intended Learning Outcomes

Our team's assessment criteria agreement is based upon a subjective analysis of work contribution to the project that relates to the Intended Learning Outcomes of the Final Year Project unit. For reference, they are as thus:

1. Apply professional practice, including active and consistent participation, delivery of technical presentations, reflection, and adherence to ethical codes of conduct as a member of a software development team
2. Apply software engineering methods and contemporary software development tools to the scoping, analysis, and design of a software system to meet client needs
3. Communicate proficiently with project stakeholders, and function as an effective member or leader of a development team in project scoping, analysis and design activities
4. Conduct a critical analysis and evaluation of aspects relevant to a software development project and justify implications for project directions

Each of these learning outcomes are to be demonstrated in the Required Documents as a part of an individual group member's submission.

2 Group Member Roles

Each group member has a self-associated role that indicates their primary area of contribution to the project.

2.1 Alex Cummaudo

Alex will focus in the following key areas, in order of importance:

1. Team Lead
2. Primary Developer
3. UX/UI Developer
4. Documentation

2.2 Jake Renzella

Jake will focus in the following key areas, in order of importance:

1. Testing Co-Lead
2. Project Developer
3. UX/UI Developer
4. Documentation
5. Filming & Media

2.3 Lachlan West

Lachlan will focus in the following key areas, in order of importance:

1. Project Developer
2. Documentation

Lachlan will develop these roles over time, and therefore his area of expertise may develop into new areas not yet listed.

2.4 Reuben Wilson

1. Testing Co-Lead
2. Project Developer
3. Research
4. Filming & Media

3 Assessment Requirements

3.1 Required Documents

3.1.1 Work Log

Each group member must produce a work log, which clearly outlines how they have allocated and spent time on the project. Work logs outline any such activity that is related to bringing the project to a state of completion.

3.1.2 Contribution Statement

Each group member must produce a project contribution statement, endorsed by each of the other group members. The contribution statement should outline every aspect of contribution toward the project made by the group member that the contribution statement belongs to.

It is the responsibility of the group to collectively decide what grade bracket each contribution statement is worth in a meeting at the end of semester.

3.1.3 Peer Review Documentation

Each group member must produce relevant peer review documents[1] for each member of the group, including themselves. Peer review provides for the ability for each group member to rate each group member against a number of key metrics.

3.1.4 Self Assessment Report

Each group member must produce a Self Assessment Report, outlining how their contributions to the overall goal of the group align to the Intended Learning Outcomes. Furthermore, each group member must use the Self Assessment Report in order to **justify the grade that they have specified that they deserve based on their contributions to the project**.

It is **compulsory** that the Self Assessment Report clearly outlines all aspects of individual contribution to the project.

3.2 Grade Breakdown

Grade breakdowns are outlined below. In order to be eligible for grading, each group member must provide all required documents, as outlined above. as well as meeting all of the criteria specified within the target grade bracket.

[1] See `https://github.com/final-year-project/documentation/wiki/Deliverables#7-peer-reviews`

3.2.1 Pass

Any group member who has done **the bare minimum** in regards to overall project contribution and progress. Such a grade would be awarded to a group member who has:

- **little to no input** in regards to project documentation, and
- **expressed no interest in voicing ideas** about actual project design and development requirements

Minimum amounts of actual project source code contribution would also contribute to such a grade being awarded to a group member.

3.2.2 Credit

Any group member who has produced **satisfactory amounts of work** in regards to project contribution.

Such a grade would be awarded to a group member who has:

- **little to moderate input** in regards to project documentation, and
- **has expressed some interest in voicing ideas** about actual project design and development requirements.

Any such group member who has shown reasonable initiative in regards to contributing to actual project source code would demonstrate eligibility for a credit.

3.2.3 Distinction

Any group member who has produced **significant amounts of work** in regards to project contribution.

Such a grade would be awarded to a group member who has:

- **moderate to large amounts of input** in regards to project documentation, and
- has expressed **significant interest in voicing ideas** about actual project design and development requirements.

Any such group member who has shown large amounts of initiative in regards to contributing to actual project source code would demonstrate eligibility for a distinction.

3.2.4 High Distinction

Any group member who has produced **significant amounts of work, showing initiative in new areas,** in regards to project contribution.

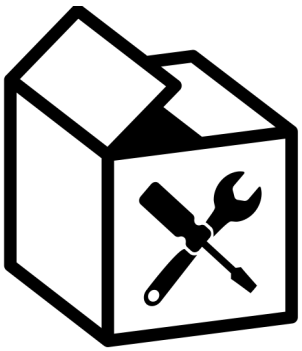
Such a grade would be awarded to a group member who has:

- **significant amounts of input** in regards to project documentation, and
- has **expressed significant interest in voicing ideas** about actual project design and development requirements.

Any such group member who has shown **large amounts of initiative** in regards to contributing to actual project source code would demonstrate eligibility for a distinction.

Additionally, any such group member who takes ownership over a particular aspect of the project development and demonstrates leadership in that field would be eligible for a High Distinction.

For example, a group member may decide to take ownership of unit testing and by doing so, commits to producing any relevant documentations in order to outline their contribution to that specific field and in order to educate the rest of the group about how unit testing is imperative to project integrity and progress.



Chapter 4

Project Tool Descriptions

Contents

1 GitHub Repositories

Figure 1: GitHub Logo

GitHub[1] is used to fork the Doubtfire API[2] and Web[3] repositories and add our additions as need be to the codebase.

It is also used to host our Portfolio and documentation as a collaborative and interactive Wiki[4], which are you reading now.

2 Slack Team Chat

Figure 2: Slack Logo

Slack is used for synchronous internal team communication. We are using the **Doubtfire Slack team**[5] to send messages to each other and digitally discuss ideas when we are not all present.

Slack is as well as the primary form of communication with our client, is is always contactable via the Slack team.

[1] See `http://github.com/final-year-project`
[2] See `http://github.com/final-year-project/doubtfire-api`
[3] See `http://github.com/final-year-project/doubtfire-web`
[4] See `https://github.com/final-year-project/documentation/wiki`
[5] See `http://doubtfire.slack.com`

3 Trello Task Management

Figure 3: Trello Logo

Trello[6] is an agile-based task management system used for asynchronous team communication and task management. We have a board[7] used to organise both our development and organisation of the final year project, where each task is represented by a card. The task progresses through the board from the leftmost column to rightmost column.

4 Celtx Storyboard Editor

Figure 4: Celtx Logo

[6]See `https://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`
[7]See `https://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`

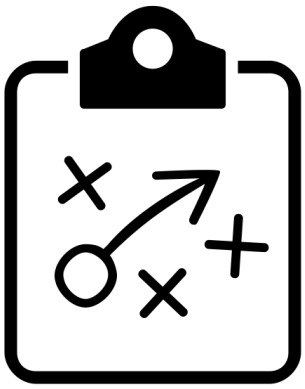
Celtx[8] is an storyboard editor used to create interactive storyboards. It will be useful for creating a rough plan of all the shots needed to create our final year project video, and uses proper terminology and toolkits adapted to the filming industry standards.

5 Invision Prototype Toolkit

Figure 5: Invision Logo

Invision[9] is a prototyping tool which can turn static images into iterative prototypes that can be tested on a device within seconds. It's rapid prototyping techniques allows for quick changes hassle free. While it is a paid service, it offers tools which are free for the team's needs.

[8] See `https://www.celtx.com/`
[9] See `https://www.invisionapp.com`



Chapter 5

Project Plan

Contents

1 Introduction

1.1 Purpose of this document

This document is designed to capture and discuss, the key processes and outcomes of the Doubtfire Helpdesk Ticketing System. This document will outline (1) how the system is designed to function, (2) how the system will effect stakeholders, as well as to assist the designers and developers of the system in the development of the project. It should be a reference of the development methodology chosen, which is outlined in the SDLC plan document.

Disclaimer: This document serves as a *guide* to the development process which will be conducted throughout this final year project. Granted the agile methodology adopted and user-centric design process, there may be significant alterations between the content outlined in this document and the final outcome itself. This is an understandable consideration considering that requirements will change as development continues, given the feedback provided from the client at the end of sprints. This has been understood by the client and product owner of the system.

1.2 Background

1.2.1 Swinburne University Programming Helpdesk

The Programming Helpdesk has been offering programming assistance to students in their first and second year of programming for many years. Over time, as the number of subjects supported by the helpdesk grows, the helpdesk has become busier, and as a result, it is very difficult for tutors working at the helpdesk to keep track of who they have seen, who still needs help and so on.

1.2.2 Doubtfire Learning Management System

Doubtfire is an open source learning management system currently in use across multiple subjects at Swinburne University of Technology and other universities in Australia. It's used by many staff and students on a daily basis. It provides students a simple and easy place to manage their unit, manage where and when they upload work, and is the place in which they receive feedback from their tutors for their submitted work.

1.3 Key Project Personnel

1.3.1 Client

The client for this project is Andrew Cain, as he oversees the running of the helpdesk and is the administrator of Doubtfire at Swinburne University. He is also a primary collaborator of Doubtfire.

1.3.2 Stakeholders

- The **Project Client**, Andrew Cain (`acain@swin.edu.au`)
- The **Project Supervisor**, Graham Farrell (`gfarrell@swin.edu.au`)
- **Teaching staff** and **students** in any unit that utilises Doubtfire as the primary learning management system used for marking and providing feedback to students.
- **Swinburne University of Technology ITS** which hosts Doubtfire and maintains server-side hardware.

1.3.3 Project Manager

Andrew Cain (`acain@swin.edu.au`) is the product owner and also a primary developer to the Doubtfire learning management system.

1.3.4 Project Members

Refer to the Group Contact Details[1] of this portfolio.

2 Terms of Reference

2.1 Goals

The Doubtfire Helpdesk Ticketing System has very clear goals which were arrived at through use of the system, and through interviews with the client. The system is intended to provide an simple, non-obtrusive way in which students can create a "ticket" when they are physically at the helpdesk, at this point tutor's should receive seamless notifications regarding their updated tickets, so that they know who and when they need to assist students who have open tickets.

[1] See `https://github.com/final-year-project/documentation/wiki/Group-Contact-Details`

The system needs to be integrated with Doubtfire for authentication and validation purposes, as well as mobile apps for the tutors so they can continue being mobile while working at the helpdesk.

The intended user group of the system are tutors and students working and seeking assistance at the Programming Helpdesk.

2.2 Objectives

1. Doubtfire should be extended to provide a in which to manage open and closed tickets created by students
2. Doubtfire should be extended to provide a way to inform all users at the helpdesk information regarding the current status of the queue (such as a projection onto a whiteboard with the current queue)
3. Doubtfire should be extended to collect analytics of the use of the Ticketing System, as well as open and closed tickets
4. Mobile Apps should be developed for tutors while working at the helpdesk so that they can have access to their tickets on the go
5. The helpdesk should be fitted with bluetooth beacons in order to ensure users of the system are physically at the location of the helpdesk

2.3 Scope

- This system will only be used at Swinburne with direct access to the current Doubtfire system.
- The system will only be used by tutors at Swinburne, with direct support from the development team if needed.
- The system will be developed alongside current development of the Doubtfire project in order to support any API needs.
- The system will be developed and supported on Swinburne provided infrastructure such as servers and devices.

2.4 Critical Success Factors

CSF1: A basic form of the system is to be rolled out to the helpdesk for use at the completion of the project.

Basic here is referring to the 'barebones' ticketing system, that is, using the new system:

- Can students continue to receive assistance at the helpdesk?
- Can tutors continue to manage and supply assistance at the helpdesk?

CSF2: Have the system running on the existing Doubtfire infrastructure, the Rails server. If this basic form of the system cannot be rolled out at the completion of the unit, the project will be deemed a failure.

CSF3: The system can collect and present analytical data to the unit conveners regarding use of the Ticketing System and the Helpdesk.

2.5 Acceptance Criteria

Upon delivery, an acceptable product will need to demonstrate the ability to perform the following functional tasks:

2.5.1 Convenors

A convenor of a unit that employs Doubtfire within the unit that they convene will be able to use the system to view analytics related to the ticketing system being used at the helpdesk. Such analytics relate to:

- Number of tickets being opened
- Number of tickets being opened by each student
- Number of tickets being resolved
- Number of tickets being resolved by each member of teaching staff working at the helpdesk
- Number of tickets opened related to a unit
- Number of tickets opened related to a specific task within a unit
- On-peak and off-peak times for when tickets are opened
- Times within the academic semester (each day and each week) when the helpdesk is the busiest and quietest

2.5.2 Students

Student's that attend the helpdesk should be able to perform the following:

- A student will be able to sign onto the Programming Help Desk upon arrival
- A student will be able to open support tickets, where each ticket they open is related to a specific task within a specific unit that employs Doubtfire as a learning management system
- A student will be able to close a ticket that they have opened

2.5.3 Tutor

Tutors employed at the helpdesk should be able to perform the following:

- A tutor will be able to sign onto the Programming Help Desk upon the beginning of their shift
- A tutor will be able to use the ticketing system to view current support tickets that have been opened by students
- A tutor will be able to delegate tickets to themselves
- A tutor will be able to delegate tickets to any other tutor currently signed on at the Programming Help Desk
- A tutor will be able to close an open ticket: (a) when the issue associated with the ticket has been resolved and (b) when a ticket has been opened and should not have been
- A tutor will have access to analytics related to the tickets and the units/tasks that the tickets are open for

3 Establishment

3.1 Process, Procedures and Standards

This is outlined in further detail under the Agile Workflow Documentation[2] and Coding Standards[3] documentation.

[2]See `https://github.com/final-year-project/documentation/wiki/SDLC-Plan`
[3]See `https://github.com/doubtfire-lms/doubtfire-web/blob/develop/CONTRIBUTING.md`

3.2 Project Environment

The physical environment of the Doubtfire Ticketing System will be at the Helpdesk (ATC620 at Swinburne University, Hawthorn Campus). The system operates in a room with desktop computers on each wall, running Windows 7, and several floating tables in the middle of the room for use with laptops.

The hardware in this room is maintained and updated by Swinburne ITS. User accounts are managed by ITS and authentication is handled by the SIMS system.

The server in which Doubtfire is run on (`doubtfire.ict.swin.edu.au`) is hosted, maintained and run by Swinburne ITS. Swinburne ITS also maintain the MySQL RDBMS which stores confidential student information, as well as the Rails infrastructure needed by Doubtfire.

Using the system requires:

- A modern desktop or mobile web browser, kept up to date. Doubtfire **does not support** Internet Explorer 9 or below, and is intended to be used by up-to-browsers such as Safari, Chrome or Firefox.
- Any desktop or laptop running a modern operating system (Windows 10, OS X 10.10+, Linux distributions released within the last two years etc.)
- Modern Android phones released in the last two years or an iPhone 5 or above.

3.3 Project Team Skills Requirements

Doubtfire is both a server-side and front-end web application, where the interface for the application is delivered through a web browser. Proficiencies and knowledge requirements are defined as thus:

3.3.1 API Server

Refer to the Doubtfire API README[4] for more information.

- **Ruby**[5]: The coding language which the server is developed in
- **Ruby on Rails**[6]: The server-side framework that allows Ruby code to run on a server

[4]See `https://github.com/doubtfire-lms/doubtfire-api`
[5]See `https://www.ruby-lang.org`
[6]See `http://rubyonrails.org`

- **PostgreSQL**[7]: The object-relational database which is used during development

3.3.2 Web Interface

Refer to the Doubtfire Web README[8] for more information.

- **JavaScript**[9] **and CoffeeScript**[10]: The coding language used to develop the front-end
- **SCSS**[11]: The styling syntax used to style Doubtfire
- **Bootstrap**[12]: A front-end framework used for styling
- **AngularJS 1.4**[13]: A front-end platform designed for building web applications

3.3.3 Additional knowledge

- **Git**[14]: A code versioning system imperative to development with regards to tracking code changes and developer contributions
- **GitHub**[15]: A website for hosting code repositories using Git tracking. Enables forking and pull-requests to merge code into the official Doubtfire codebase[16].
- **RESTful architectures**[17]: Representational state transfer architecture styling which most modern web-applications are based on.
- **Socket.IO**[18]: Socket-based JavaScript library used to build realtime web applications.

[7] See `http://www.postgresql.org`
[8] See `https://github.com/doubtfire-lms/doubtfire-web`
[9] See `https://www.javascript.com`
[10] See `http://coffeescript.org`
[11] See `http://sass-lang.com`
[12] See `http://getbootstrap.com`
[13] See `http://angularjs.org`
[14] See `https://git-scm.com`
[15] See `http://github.com`
[16] See `http://github.com/doubtfire-lms`
[17] See `https://en.wikipedia.org/wiki/Representational_state_transfer`
[18] See `http://socket.io`

4 Activities, Deliverables and Capital Resources

4.1 Deliverables

A functioning extension of Doubtfire which will enable students and tutors to create and manage tickets. It will be completed using the Git development system currently outlined in the contributing documentation and merged into the live Doubtfire system by the client. This should include the analytics system.

Two functioning mobile apps developed and uploaded to the respective Apple App or Google Play stores which allows tutors to download and run the apps.

4.2 Activities and Tasks

There are a number of activities that the group members must embark on in order to ensure that the project's development and overall delivery a success. The key activities that have been identified as critical to the project's success are outlined and defined in detail below.

4.2.1 Research

Research is absolutely imperative in terms of understanding the need for the project and the domain for which the project is being developed for. Research, when done correctly, provides for a concrete foundation on which to build the project and to ensure that it is stable throughout the entire development cycle.

Without researching the project's target domain and the functionality that it will offer, the development would be based upon a loose construction of ideas that vaguely resemble what purpose the product is supposed to serve. Not only does research provide for a solid foundation on which to develop the project, it aids in developing better problem solving skills, critical thinking measures, confidence in what you're developing as well as project driven motivation.

4.2.2 Development

The development activity is where the knowledge gathered from the research phase is applied and constructed into a meaningful representation of what the project is supposed

to represent. Throughout the development phase, each group member will be completely focussed on implementing a seamless solution.

4.2.3 Testing

A test-driven development (TDD) methodology will be adopted. This will include testing each and all of the different tool sets that, combined, produce the overall functionality of the ticketing system.

It is extremely important to conduct rigorous testing in order to ensure that the product is working exactly as is intended. It is also important to identify any issues that the system may have during the testing phase so that these may be corrected.

4.2.4 Documentation Authoring

After the project has been developed and tested rigorously and meets a standard of functionality that all of the group members can agree upon, documentation needs to be developed.

During this stage, user manuals and technical documentation will be authored in an interactive fashion (i.e., via a Git wiki) to provide for a complete dissection of the system and how it all works.

4.2.5 Submission for Critical Analysis

This task involves submitting the final project with all accompanying documentation for critical analysis from an individual(s) separate to the group dynamic. It is the intention of the group to have any such individual walk away from analysing the project feeling completely satisfied in the final product and all of the documentation provided with it.

4.3 Resources

4.3.1 Organisation Structure

The organisation structure of the helpdesk is listed in hierachy as thus:

1. **Helpdesk Administrator**, Andrew Cain
2. **Helpdesk Subject Convenors**:
    - Andrew Cain, Introduction to Programming

- Alan Colman (acolman@swin.edu.au), Creating Web Applications
- Chris McCarthy (cdmccarthy@swin.edu.au), Object-Oriented Programming

3. **Helpdesk Subject Tutors**
4. **Helpdesk General Support Staff**
5. **Helpdesk Volunteers**

4.3.2 Development Equipment

UNIX-based systems to develop the system is required as Doubtfire does not support development on Windows. In addition, iOS applications must be developed using the OS X operating system, which will require the procurement of iMacs. These can be procured through Swinburne ITS.

To test the mobile applications, an iPhone(s) will be required for the iOS application and several Android devices will be required. Lastly Bluetooth BLE beacons used to clock staff on and off will be required. These can be procured through scholarships provided by Swinburne for final year students.

5 Risk Analysis

5.1 Risk 1

Possible Issue: Misunderstanding or poor interpretation either via electronic or verbal communication
Chance of Occurrence: High
Severity of Detriment to Group Ambitions: Little to none
Preventative Methods:

1. Ask clear and concise questions
2. Communicate any ideas as soon as they dawn
3. Document clear notes
4. In the event that something is not clearly communicated, make the issue known

5.2 Risk 2

Possible Issue: The group members all have different opinions in regards to how a particular problem should be approached
Chance of Occurrence: High
Severity of Detriment to Group Ambitions: Low to Medium
Preventative Methods:

1. Elect to resolve the opinion dispute by casting a majority vote
2. Consult the project supervisor for their opinion

5.3 Risk 3

Possible Issue: Task delegation becomes an issue because no group member elects to take on responsibility
Chance of Occurrence: Medium
Severity of Detriment to Group Ambitions: Low to Medium
Preventative Methods:

1. Refer to the elected responsibilities
2. Consult project supervisor for indisputable delegation authority

5.4 Risk 4

Possible Issue: Absent from Lectures
Chance of Occurrence: Medium
Severity of Detriment to Group Ambitions: Medium to High
Preventative Methods:

1. Always inform group members if your attendance is in jeopardy
2. Understand that you're an integral part to the group's success

5.5 Risk 5

Possible Issue: Absent from arranged group meetings
Chance of Occurrence: Low
Severity of Detriment to Group Ambitions: Medium to High
Preventative Methods:

1. Always inform group members if your attendance is in jeopardy
2. Understand that you're an integral part to the group's success
3. Clearly communicate any commitment issues with fellow group members

5.6 Risk 6

Possible Issue: Development timelines ignored/not met
Chance of Occurrence: Low
Severity of Detriment to Group Ambitions: Severe
Preventative Methods:

1. Understand that the group is relying on a strong work ethic to produce exceptional results
2. Constantly check your delegated tasks and review deadlines
3. Understand your role within the group
4. Understand that without submissions, the whole group suffers

5.7 Risk 7

Possible Issue: Document availability issues
Chance of Occurrence: Low
Severity of Detriment to Group Ambitions: Severe
Preventative Methods:

1. Utilise a single medium for document sharing and collaboration
2. Ask questions and know which tasks are delegated to each group member

5.8 Risk 8

Possible Issue: Transportation issues
Chance of Occurrence: Low to Medium
Severity of Detriment to Group Ambitions: Low
Preventative Methods:

1. Use punctuality to ensure that your transport is not jeopardised
2. In the event that transport is absent, communicate your situation with fellow group members

5.9 Risk 9

Possible Issue: Delegated task not completed
Chance of Occurrence: Low
Severity of Detriment to Group Ambitions: Severe
Preventative Methods:

1. Offer a group communication with the group member who has not produced material.
2. Involve the subject convenor and arrange a remedy.

6 Schedule

6.1 Overview and Timeline

An overview of the semesterly schedules are outlined in Figure 1. The team plans to adopt a fortnightly sprint, surrounded by a week of sprint planning prior to the sprint beginning and a week of a sprint retrospective. Further information regarding the workflow is outlined in the Agile Workflow Summary[19].

The overarching plan for deliverables will occur during a Sprint Retrospective. Within the Sprint Retrospective, work from the previous sprint is shown to the client, which serves as a feedback loop for the previous two-week iteration.

[19]See `https://github.com/final-year-project/documentation/wiki/SDLC-Plan`

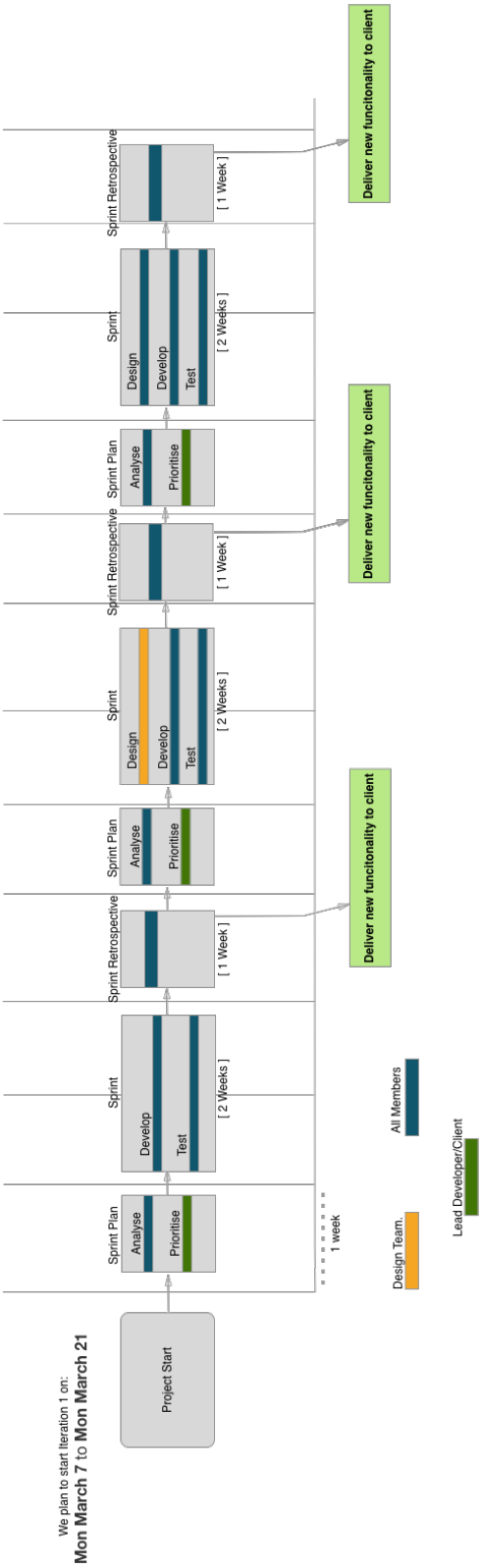


Figure 1: Semesterly schedule, outlining Semester 1

Thus, the general principle will be to:

1. Plan a sprint, deciding which tasks need to be allocated
2. Work on each task over a two-week sprint
3. Deliver new functionality to the client during the retrospective week
4. Improve upon the product given the client's feedback and work on this for the following week.

Whilst the dates are time-fixed, the durations of sprints, sprint planning and sprint retrospectives are *dynamic*, and so dates may not be adhered to concretely since they only act to serve a guide. For example, there may be a time where a two-week sprint is simply not enough time given the time allowed, and thus, the two-week schedule may be extended to a three or four week schedule, reducing planning and retrospective time.

This is chiefly due to external dependencies on the project, especially since the technical requirements will be forever changing with client demands.

6.2 External Dependencies

6.2.1 Client and Helpdesk Manager

As Andrew Cain is the lead developer for Doubtfire, and he is also the manager of the Programming Help Desk, his decisions, choices and influences largely sway the direction of the project.

The team will be working closely with Andrew to ensure that he and the team always stay on the same page in terms of the project. In an agile context, the team will conduct weekly 'catch up' meetings which act, serving as the stand-ups the team involves the client in.

6.2.2 Other Studies

Other final year subjects have large amounts of work during assessment times, which will sometimes overlap with development of the project.

6.2.3 Personal Life

This includes general physical and mental health of each team member, which is obviously more important than any work.

Table 1: Breakdown of estimated and actual hours over the semesters

Team Member	S1 Estimation	S1 Actual	S2 Estimation
Alex	132	135	132
Jake	132	146	132
Reuben	132	50.25	132
Lachlan	132	70.5	132

6.3 Assumptions Made

Completion of the project assumes that the relevant Swinburne services will aid us. ITS is to provide software and hardware required to run the system, and will also provide developmental resources as outlined in the previous sections. It is also expected that Swinburne will aid the team by providing the team with some suitable development room, as well as development hardware.

Another assumption made is that the helpdesk will still operate 'as-is'; the need for the project is still relevant throughout the year. There will be no unexpected downtime of the helpdesk or indeed the University.

7 Budget

The breakdown of hours have based on worklog hours. Both the semesterly estimates, and actual recordings are for Semester 1 are given in Table 1. The sprint hours allocated for this semester have been indicated also, as visualised in Figure 2 from data shown in Table 2.

Estimates for the project have been calculated as thus:

1. 96 hours minimum allocated for project work, either performed individually or together
2. 24 hours minimum allocated for team and client meetings
3. 12 hours minimum allocated for supervisor meetings

This gives a total of 132 hours per semester.

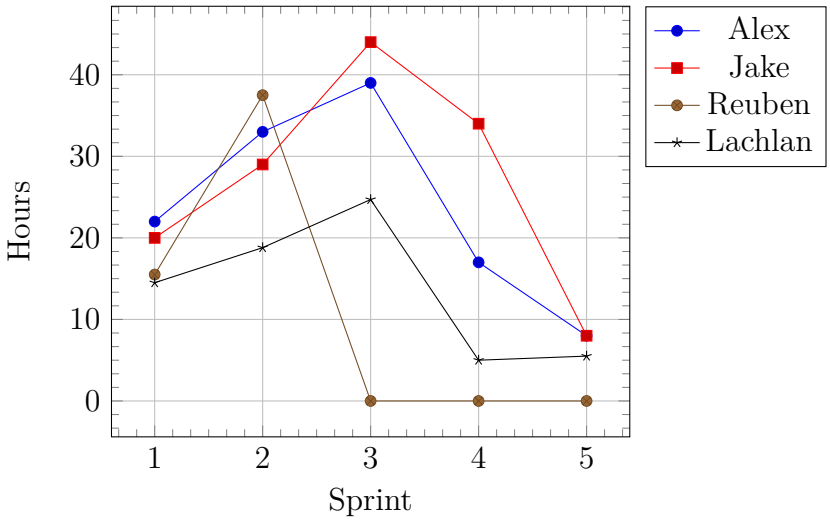
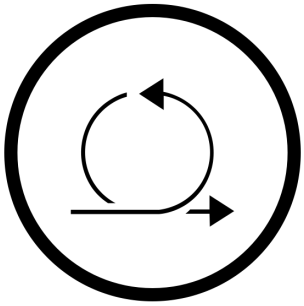


Figure 2: Graphical representation of hours per sprint

Table 2: Breakdown of recorded hours for each sprint

Sprint	Alex	Jake	Reuben	Lachlan
1	22	20	15.5	14.5
2	33	29	37.5	18.8
3	39	44	0	24.7
4	17	34	0	5
5	8	8	0	5.5



Chapter 6

Agile Methodology Workflow

SDLC Plan

Contents

1 Development Framework

The team has decided to adopt a **Scrum framework** in our **Agile approach** to deliver this software.

2 Sprints

The team will adopt multiple sprints throughout the duration of the project, whose dates and goals are defined by a group decision and with the client.

2.1 Sprint Dates & Goals

The following lists each of the Sprints, their dates and intended goals. This list will be updated as new sprints are devised throughout the development lifecycle.

1. **March 7 to March 21** - This sprint was decided in the March 7 meeting[1] with our client and is intended to "consist of small bug fixes or enhancements that will help the team familiarise themselves with Doubtfire's codebase"
2. **March 21 to April 4** - This sprint builds upon the first sprint whilst concurrently working on requirements documents and assessment criteria for the project. Jake and Reuben will take on unit testing as it has been neglected in the past for the Rails API and will investigate methods to test it.
3. **April 4 to May 4** - This sprint is the largest sprint of the semester. It focuses on the team dedicating time to shift their resources on a majority of requirements analysis for the system, including prototyping and architecture decisions. Like Sprint 2, it is expected that team members also work on tasks as need be for practise and familiarisation of the Doubtfire codebase
4. **May 4 to May 23** - This sprint focuses on the film which needs to be shot by the presentation on May 23 in Week 11. In addition, this sprint will focus on the majority of the requirements documentation that needs to be properly thought about, such as use case descriptions.
5. **May 23 to May 30** - This last sprint wraps up the first semester as the team prepares their portfolio for submission and assessment. It should include a thorough

[1] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#re-4-spiking-doubtfire-for-the-next-few-weeks`

review of the documents produced thus far, and make additional 'last-minute' changes if need be.

3 Development workflow

3.1 High level overview

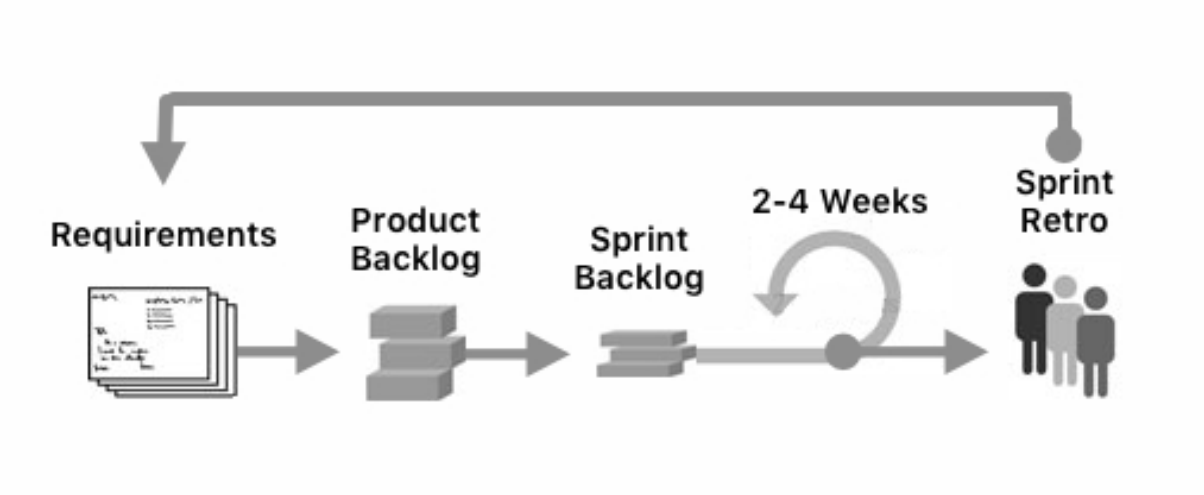


Figure 1: High level overview of the workflow

The high-level overview of the workflow is an adapted standard scrum process that better suits the needs of the project:

1. The **requirements** of the project is devised with the client in **ongoing client meetings**. This means that requirements can be added on an ongoing basis throughout the project
2. Requirements are fed into the **project backlog**, that is the backlog of the entire Doubtfire system, as tasks
3. Specific tasks are chosen to be worked on a sprint into a **sprint backlog**
4. The sprint lasts for 2-4 weeks, where tasks from the sprint backlog are eventually all completed
5. A **sprint retrospective** is run amongst both the team and the client, where feedback from the sprint is eventually fed back into initial the requirements

There are no daily stand-ups like the standard scrum framework mandates. This is due to time and constraints with the group, as only one or two meetings are possible throughout the week. The team aims for a meeting together once a week, and a meeting with the client on a fortnightly basis.

To slimline the workflow, the sprint review process is merged into the sprint retrospective, meaning the retrospective is conducted with the team and client such that the client gets a feeling of how the team is progressing through tasks *and* can suggest changes to the product in one go.

The workflow enhances Doubtfire's **continuous integration**[2] as each changes will be pushed to the `develop` branch, and eventually `master` branch (meaning that as tasks get completed, the code changes will go live to production).

3.2 Trello Boards

As described by the Project Tools[3] document, Trello is being used to manage tasks. The Trello Workflow adapts to the high level overview in the form of variant boards and columns. Each board and its intended workflow is outlined below in further detail.

3.2.1 Doubtfire Backlog Board

The **Doubtfire Backlog Board**[4] is the *Product Backlog* for Doubtfire. It aims to outline all tasks in Doubtfire's backlog horizontally. Each column is described as thus:

1. **Fresh Ideas** - New tasks that the product owner thinks of will be added here. But those tasks should be be moved into one of the other lists on this board as soon as possible, and therefore this list should be kept as empty as possible.
2. **Quick and Easy**
3. **Top Priority** - Tasks that need to be completed ASAP, such as critical bugs
4. **High Priority** - Tasks that need to be completed which have high importance, such as bugs
5. **Medium Priority** - Tasks that should be completed soon, such as new features or enhancements

[2] See `https://en.wikipedia.org/wiki/Continuous_integration`

[3] See `https://github.com/final-year-project/documentation/wiki/Project-Tools#trello-task-management`

[4] See `https://trello.com/b/0uh6AZdu/doubtfire-backlog`

Figure 2: The Doubtfire Backlog Board

6. **Low Priority** - Tasks that would be nice to have done eventually, such as small UI beautifications
7. **One Day** - Ideas that would be great to have *one day* if we had the time, essentially tasks that are currently too out of scope
8. **Maybe...** - Ideas that are still being considered

The Doubtfire Backlog Board is maintained by the Product Owner (i.e., the client)—the team does not have access to add new tasks here unless it is approved by the Product Owner.

3.2.2 Helpdesk Ticketing System Board

The **Helpdesk Ticketing System Board**[5] is the board that outlines all tasks for the team's project.

This board not only contains the **Sprint Backlog** tasks, but also meta tasks related to administration and assessment that needs to be done (such as organising meetings with

[5]See `https://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`

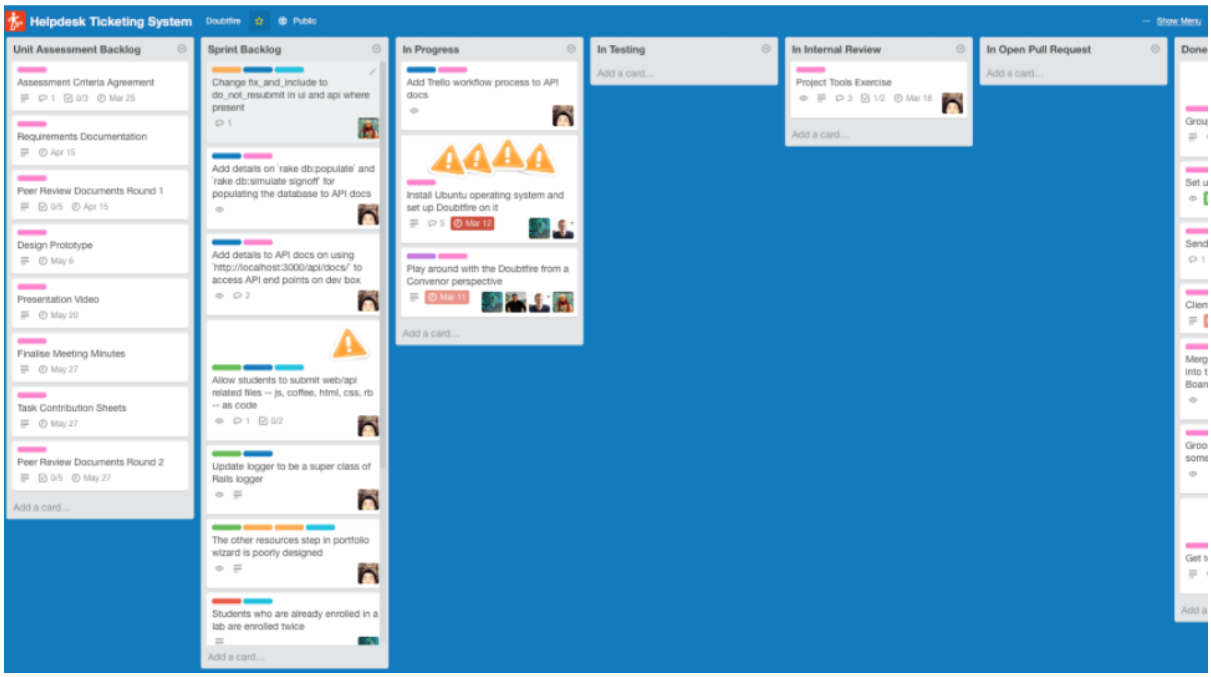


Figure 3: The Helpdesk Ticketing System Board

clients, installing required software etc). This will help the project manager organise which tasks the team needs to not only on a *developmental* basis, but also on an *assessment* and *administration* basis for the final year project unit.

The board is organised into several columns:

1. **Unit Assessment Backlog** - The backlog of tasks that are required by the final year project unit deliverables[6]
2. **Sprint Backlog** - Tasks involved over the upcoming time-fixed sprint moved from the Doubtfire Backlog board - essentially the current sprint backlog
3. **In Progress** - Tasks that are currently in progress. These tasks **must be assigned to whoever is working on them**
4. **In Testing** - Where relevant, tasks move into this column if unit or integration testing is needed on that task (e.g., a new API endpoint should have unit tests written). **These tasks should be assigned to whoever is writing the tests for the task**

[6]See `https://github.com/final-year-project/documentation/wiki/Deliverables`

5. **In Internal Review** - Tasks that are to be reviewed internally by another team member. **These tasks should be reassigned to the reviewer**
6. **In Open Pull Request** - Tasks that have been put into a Pull Request and assigned to a product owner for external code review. **These tasks should be reassigned to the external code reviewer**.
7. **Done** - When the task is merged into the `develop` branch (the Pull Request has been closed).

This board is maintained by the project manager of the team, and also updated by team members as they progress through their allocated tasks.

3.2.3 Trello Workflow in a Sprint

This workflow is used to guide team members on how to use Trello for their day-to-day activities whilst working on Doubtfire.

The workflow also complements the Doubfire Git Workflow[7], meaning that all changes made will be pushed into the primary `develop` branch on the Doubtfire product workflow for improved continuous integration.

3.2.3.1 Prepare cards for a sprint

In this step, team members will go to the Doubtfire Backlog Board[8] and find (or create with the Product Owner's permission) tasks that are relevant for the current sprint.

Once they have found a card, they can move it directly to the Helpdesk Ticketing System Board[9]'s Sprint Backlog column. To do this, they can click on a card and them move it:

3.2.3.2 Progressing the card

Once the card is in the backlog on the Helpdesk Ticketing System Board, it will be progressed throughout the board (moved toward to the right hand side of the board) to describe its process by dragging and dropping it between columns:

[7]See `https://github.com/doubtfire-lms/doubtfire-api/blob/develop/CONTRIBUTING.md`
[8]See `https://trello.com/b/0uh6AZdu/doubtfire-backlog`
[9]See `https://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`

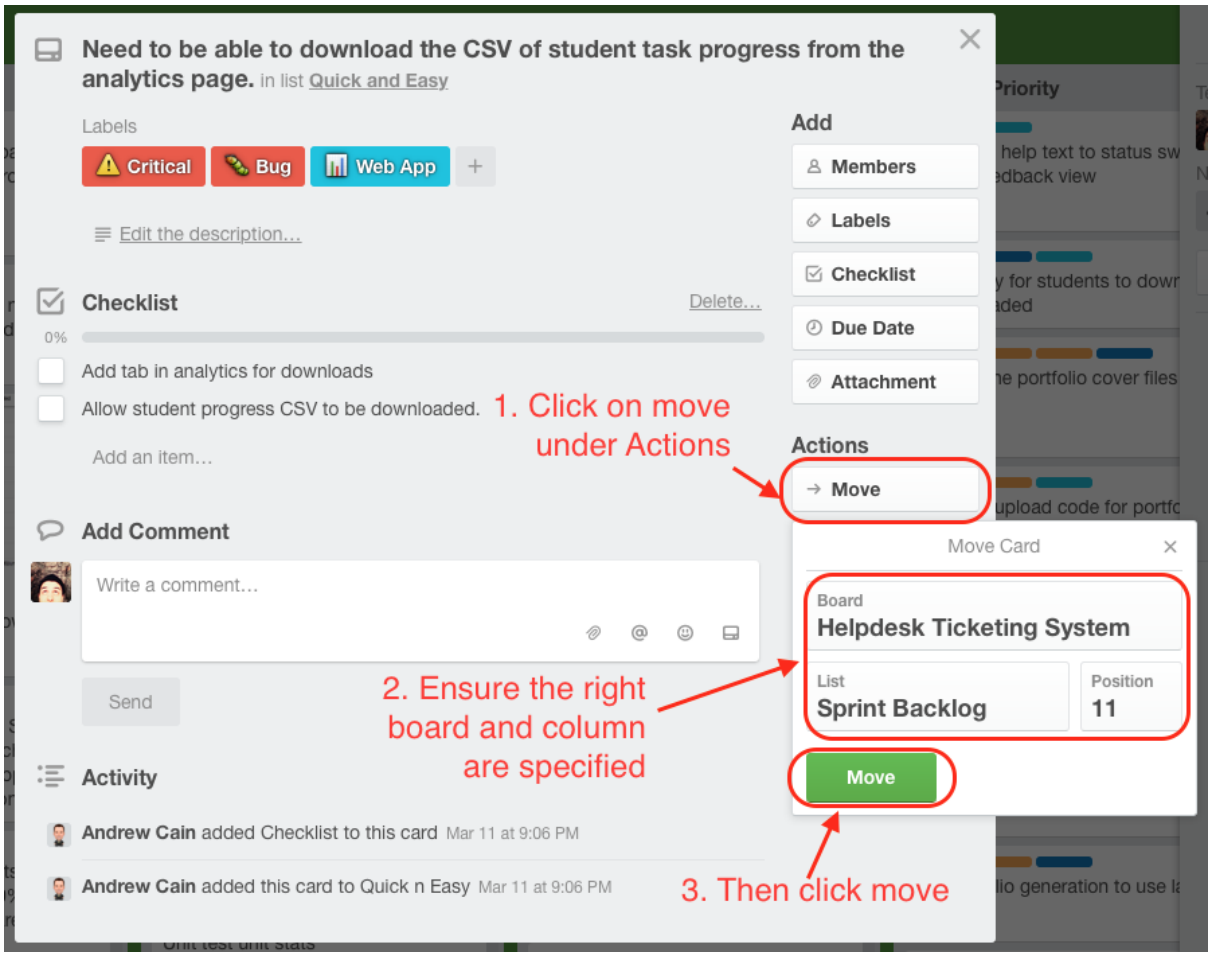


Figure 4: Moving a card to the right board

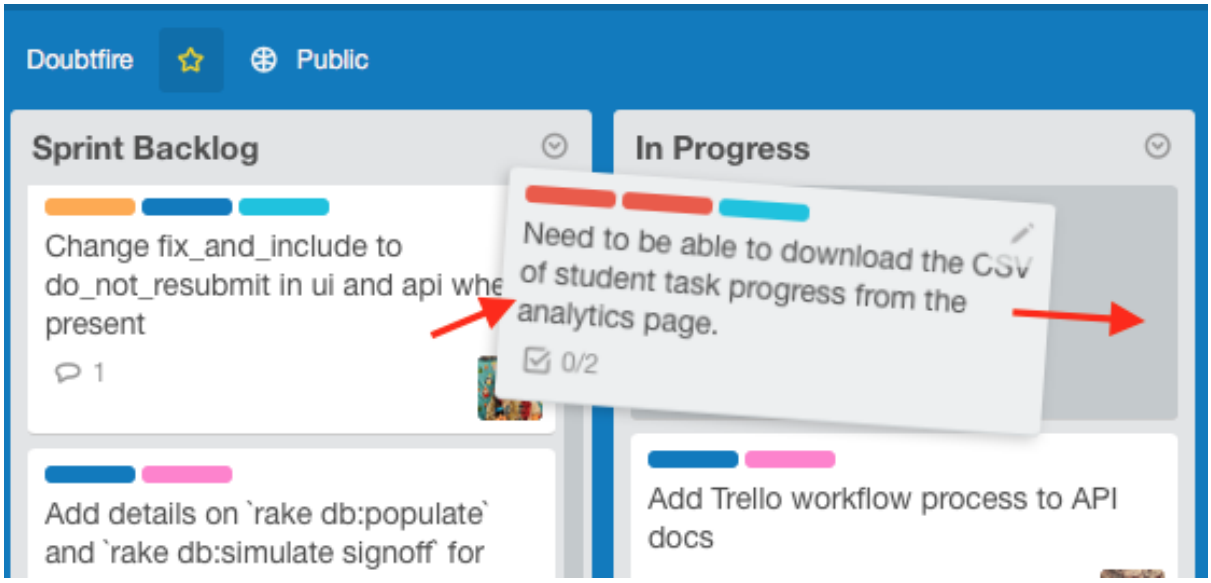


Figure 5: Dragging and dropping the card

3.2.3.2.1 In Progress

When the card is ready to start on, progress the card from the **Sprint Backlog** to the **In Progress** column.

It needs to be assigned to whoever is working on the card. To do so, click on the card and assign the task to a team member:

3.2.3.2.2 In Testing

If the card requires testing to be done, progress the task from **In Progress** to **In Testing**.

3.2.3.2.3 In Internal Review

When the person working on the card thinks the card is ready, they should run a **code walkthrough** with someone else in the team.

Progress the task from **In Testing** or **In Progress** to **In Internal Review** and sit down together with the team member. Walk the other team member through both the *functionality* changes added, as well as the *code* that has been added. This ensures for optimal product quality and code quality by having a second set of eyes look over the code.

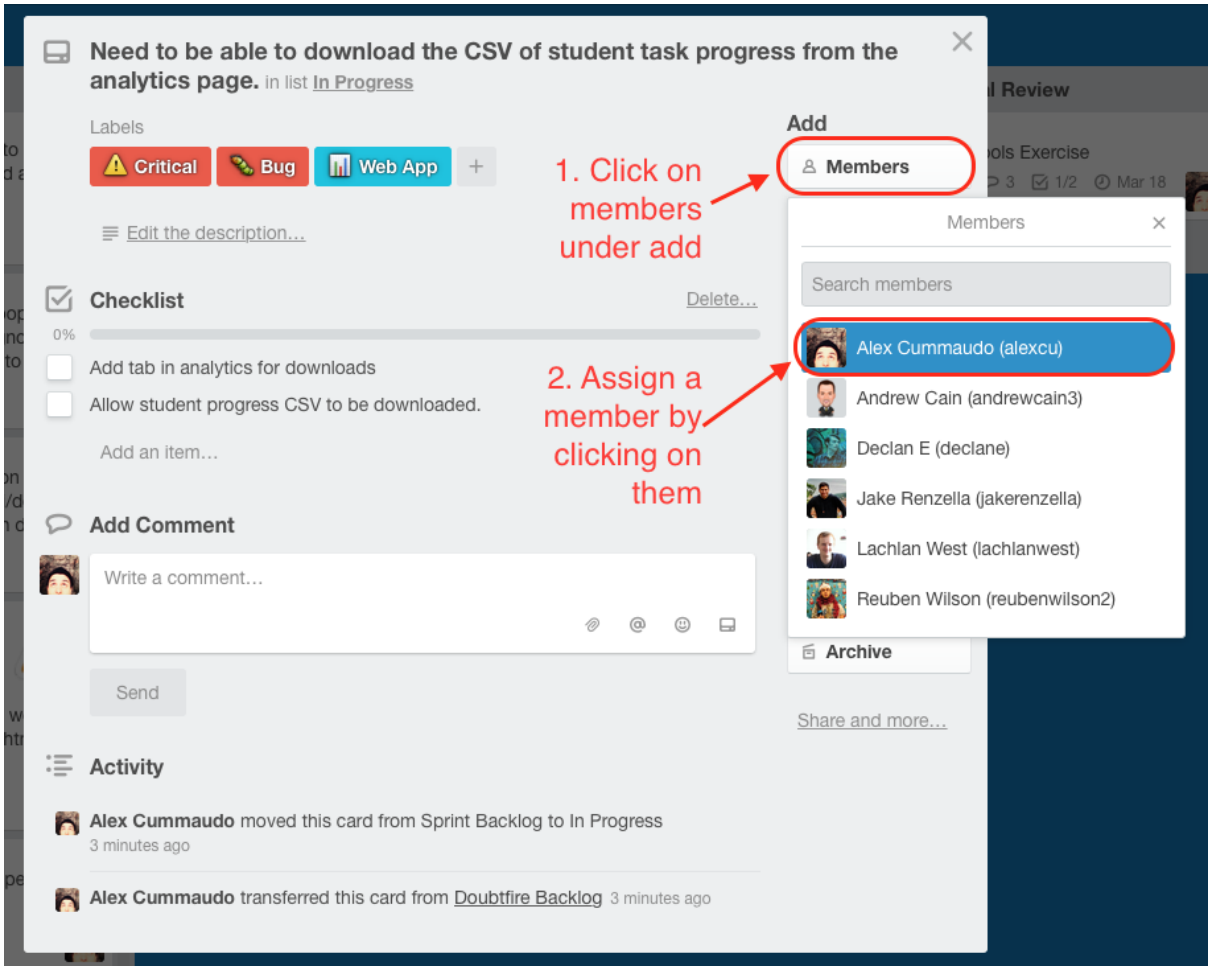


Figure 6: Assign a member

When the code review is over, remove the other team member from the members list of the card.

3.2.3.2.4 In Open Pull Request

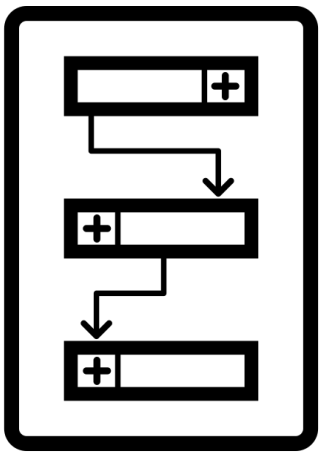
As through the Doubtfire Git Workflow[10] a Pull Request should be submitted to the Product Owners (i.e., the `doubtfire-lms` repository) when ready for external review.

Progress the card to **In Open Pull Request** from **In Internal Review** when the pull request has been submitted and assign the Product Owner (e.g., Andrew Cain) to the card (if applicable).

3.2.3.2.5 Done

Progress the card from **In Open Pull Request** to **Done** only once the changes made from the card have been merged and the Pull Request is closed. The assigned task member can be removed from the card at this point.

[10] See `https://github.com/doubtfire-lms/doubtfire-api/blob/develop/CONTRIBUTING.md#4-submitting-a-pull-request-pr-to-the-upstream-repository`



Chapter 7

Technical Requirements

Contents

1 Goals & Objectives

Whilst reading this documentation, it is important to keep the following goals and objectives in mind:

The Doubtfire Helpdesk Ticketing System will provide:

- A way to improve efficiency of helping students
- A way for tutors to track which students need help
- A way to manage tutor clock-on times
- A way for convenors to see how much their students utilise the helpdesk and at what times
- A way for convenors to to see how their tutors are clocking on at the helpdesk

Refer to the extended planning documentation for more on this.

2 Entities

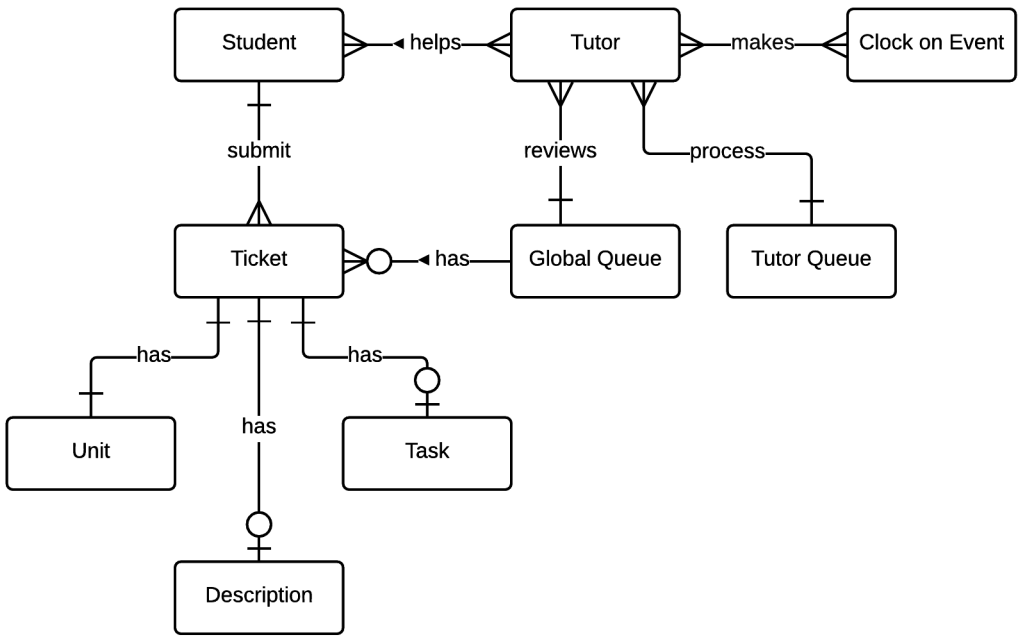


Figure 1: Entity Relationship Diagram highlighting fundamental entities

Figure 1 shows a basic ERD, highlighting the fundamental entities that the helpdesk ticketing system works with.

There is a distinction between a **tutor's queue** and the **global queue**.

A **tutor's queue** is a group of a number of students that a tutor rotates through as he/she helps those students at the helpdesk. It is sorted by the **inverse** time the ticket was last reviewed by the tutor (i.e., last reviewed a while ago or never reviewed at all first, just reviewed a few seconds ago at the end etc.)

A **global queue** is a list of *all unallocated tickets* that have been submitted at the helpdesk. Tutors working at the helpdesk aim to keep this global queue as minimal as possible; when a new ticket comes to the global queue, tutor's may:

1. accept a new ticket, which will move that ticket to the tutor's own queue or,
2. refer that ticket to another tutor, which will move that ticket to the referral's queue.

3 Use Cases

Refer to a summary of use cases in Figure 2.

3.1 Students

3.1.1 Submit a ticket

3.1.1.1 Primary Use Case

Step 1. Student signs into Doubtfire
Step 2. Student selects Helpdesk from header
Step 3. Student selects unit they want help with
Step 4. Student submits the ticket. Doubtfire adds their ticket to the **global queue**
Step 5. Student views an estimate of wait time

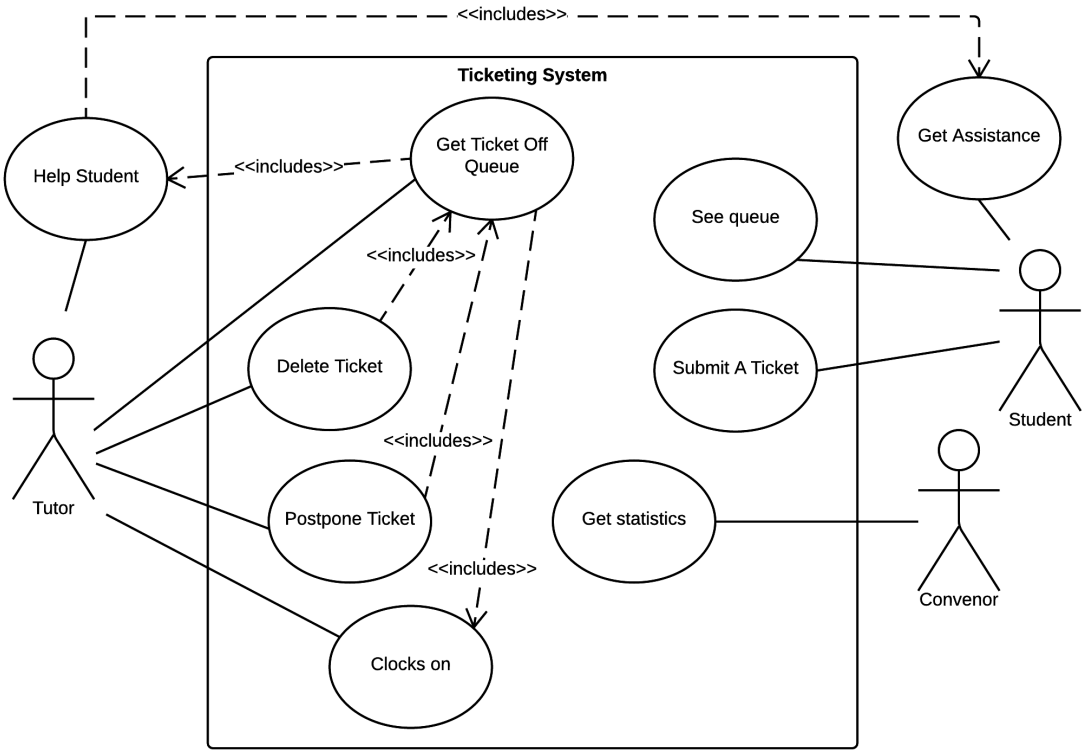


Figure 2: Summary of use cases and interacting actors of the system

.

3.1.1.2 Alternate Use Cases

Student doesn't have a computer

Step 1a. Student goes to instructor PC
Step 1b. Student enters in their student ID
Step 1c. Continue from (3)

Helpdesk ticket queue is overloaded

Step 2a. Student is given a visual notice that they might have to wait a while to get help
Step 2b. Student *optionally* cancels the process
Step 2c. Student *optionally* continues from (3)

3.2 Tutors

3.2.1 Clocking On

3.2.1.1 Workflow Diagram

Refer to the workflow diagram provided in the Appendix, Figure 4. This diagram outlines the basic preconditions required for the use cases to be accepted.

3.2.1.2 Primary Use Case

Step 1. Tutor approaches the vicinity of the helpdesk
Step 2. A push notification is received on the tutor's smartphone. See Figure 5.
Step 3. Tutor accepts the push notification and they are clocked on. See Figure 6.

3.2.1.3 Alternate Use Cases

Tutor isn't yet signed into the helpdesk app **or** *bluetooth is disabled*

Step 2a. No push notification is sent
Step 2b. Tutor follows sign in process as indicated in workflow diagram above

Push notifications are disabled **or** *tutor dismisses the notifica tion*

Step 3a. Tutor opens the Helpdesk app on their smartphone and manually clocks on. Refer to Figure 7.

3.2.2 Getting the next ticket off the global queue

3.2.2.1 Primary Use Case

Step 1. Tutor taps the name of the student
Step 2. Tutor accepts the ticket and it is added to the top of their queue

3.2.2.2 Alternate Use Case

Tutor has too many students at the moment and wants someone else to see the s tudent

Step 2a. Tutor refers the ticket to another tutor
Step 2b. Tutor selects a tutor from a list of tutors currently clocked on at the helpdesk
Step 2c. Doubtfire adds that ticket to the queue of the selected tutor

3.2.3 Reviewing the topmost ticket from the tutor's queue

3.2.3.1 Primary Use Case

Step 1. Tutor taps the name of the student
Step 2. App shows details about that ticket
Step 3. Tutor marks the details about the ticket as resolved
Step 4. Ticket is removed from their queue and is purged

3.2.3.2 Alternate Use Case

Tutor indicates that they'll come back later and review the student late r on

Step 3a. Tutor marks the ticket and says that they'll come back later
Step 3b. Ticket is pushed down to the end of their queue; time last saw is updated to now.

Student isn't physically present

Step 2a. Tutor postpone's the ticket; run use case above.

4 High Level Architecture Diagram

A monolithic architecture is to be discouraged whilst developing the helpdesk ticketing system. This is because the one API will keep expanding and become far too big when it doesn't necessarily need to be. This may lead to the APInest becoming too unmaintainable.

In addition, some of the technologies considered, namely Redis and other realtime application frameworks (Socket.IO), cannot be implemented into the existing Rails architecture. An independent microservice will be created to handle this instead.

Refer to Figure 3 for more.

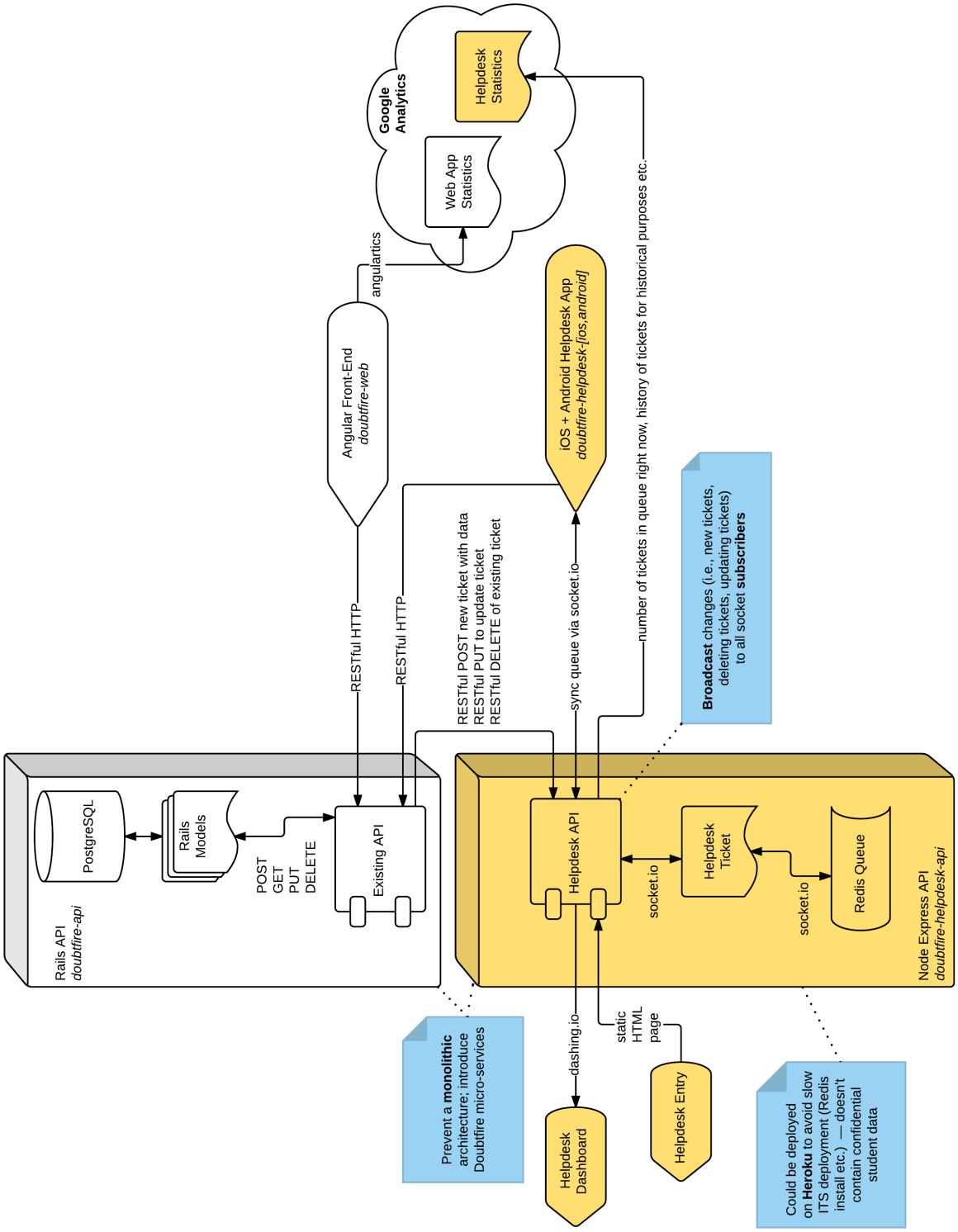


Figure 3: High Level Architecture Diagram

A Supporting Use Case Images

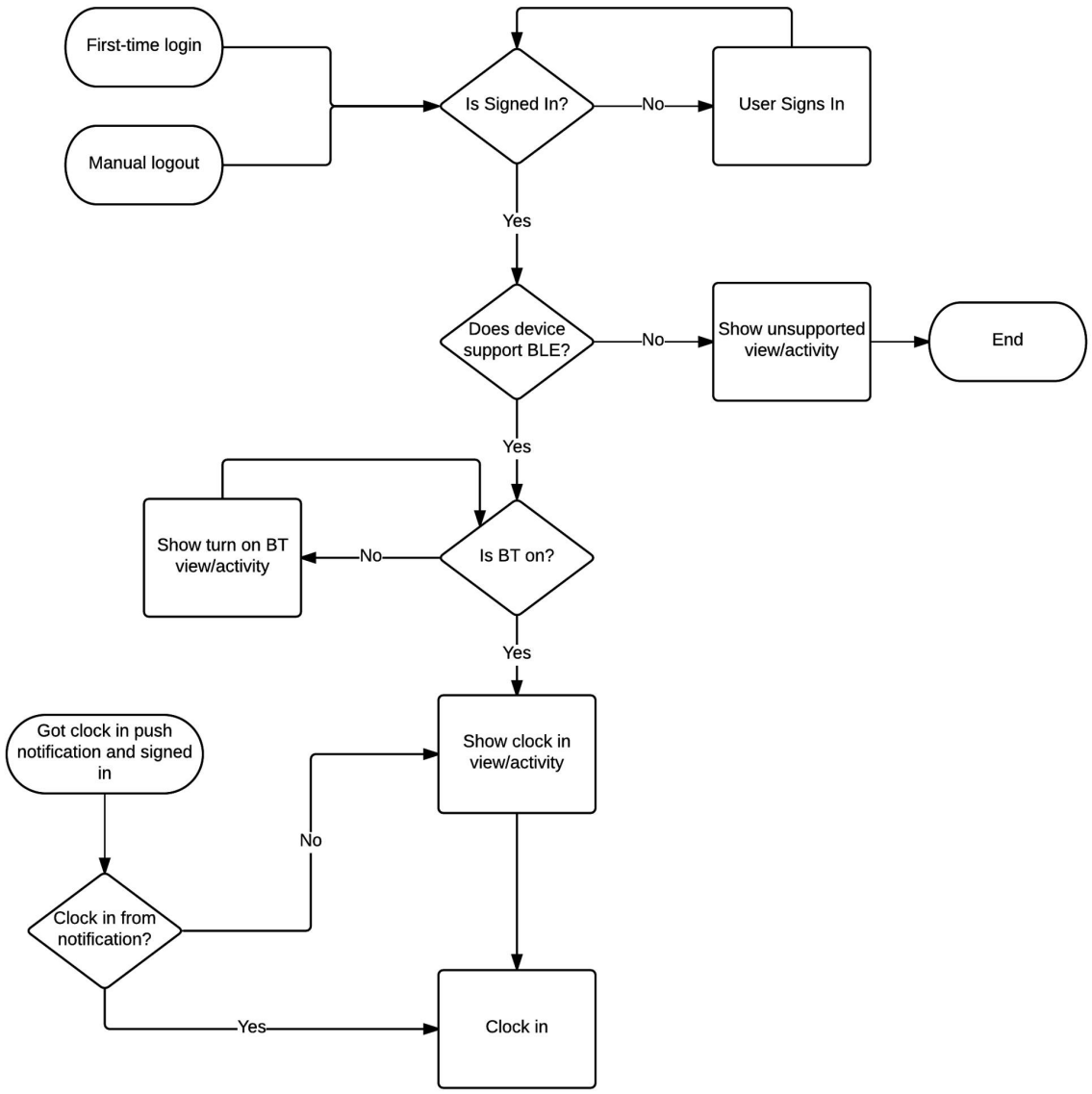


Figure 4: Workflow diagram for clocking on

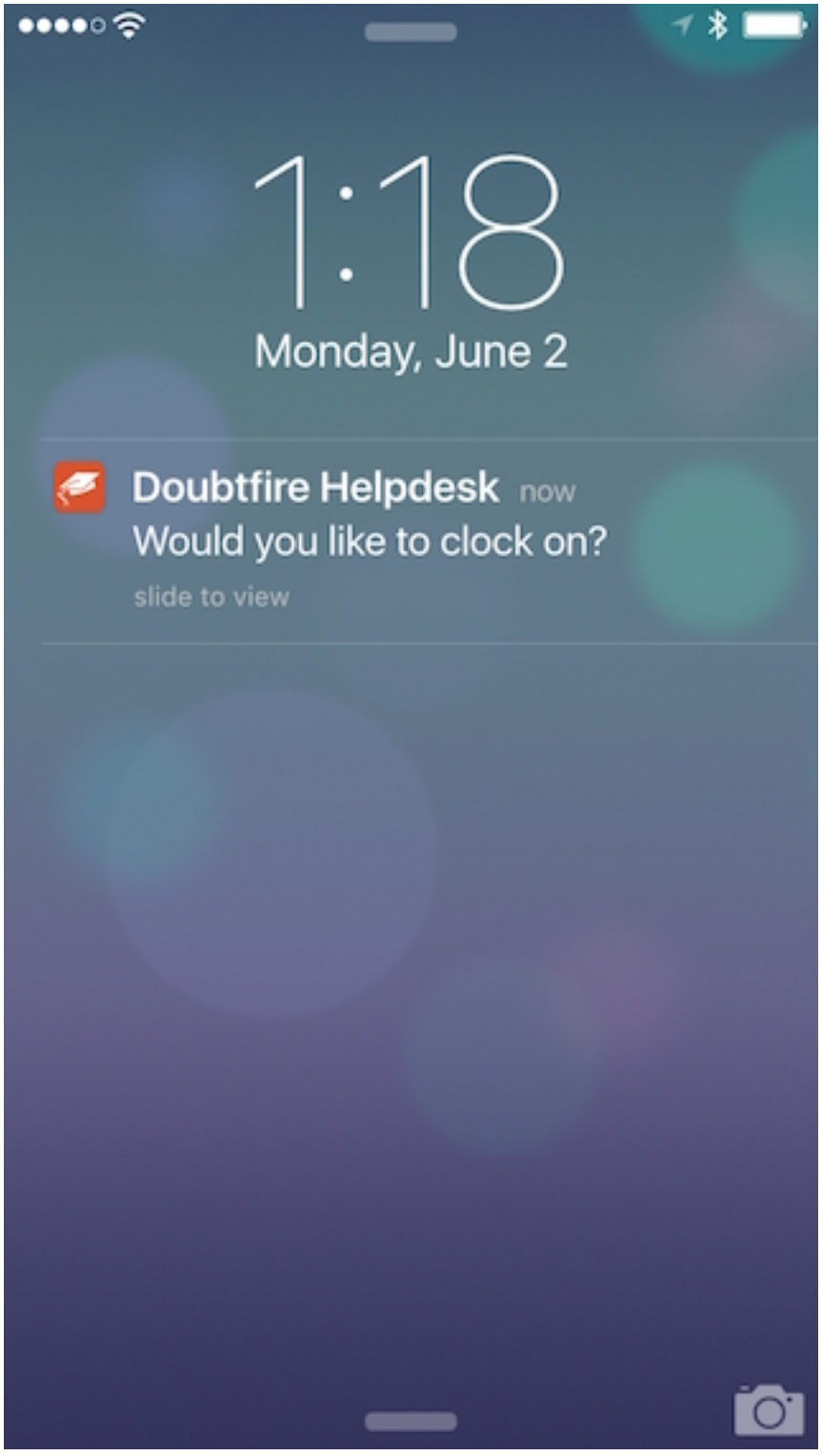


Figure 5: Push notification to clock on

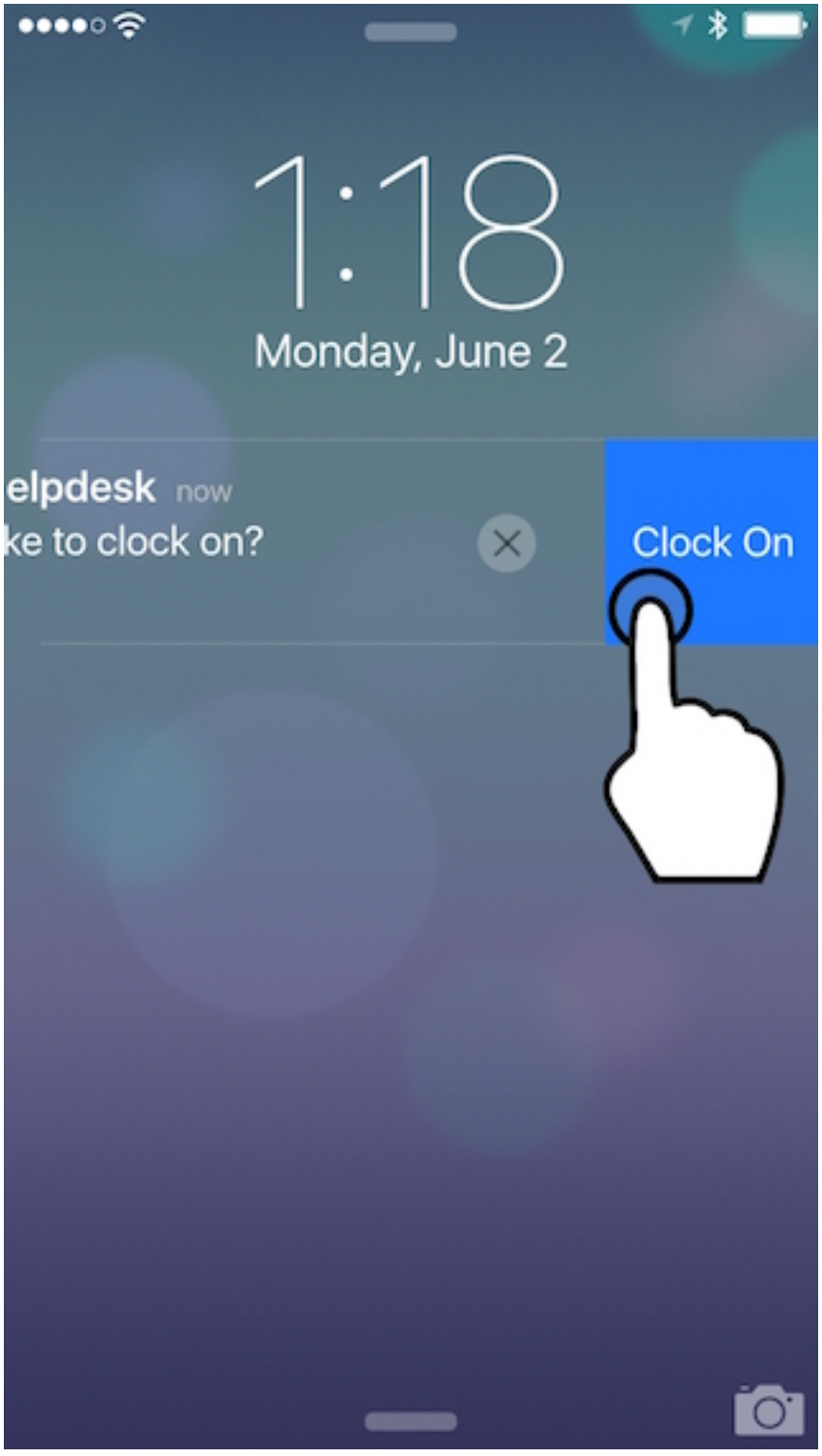


Figure 6: Clock on accept via notification

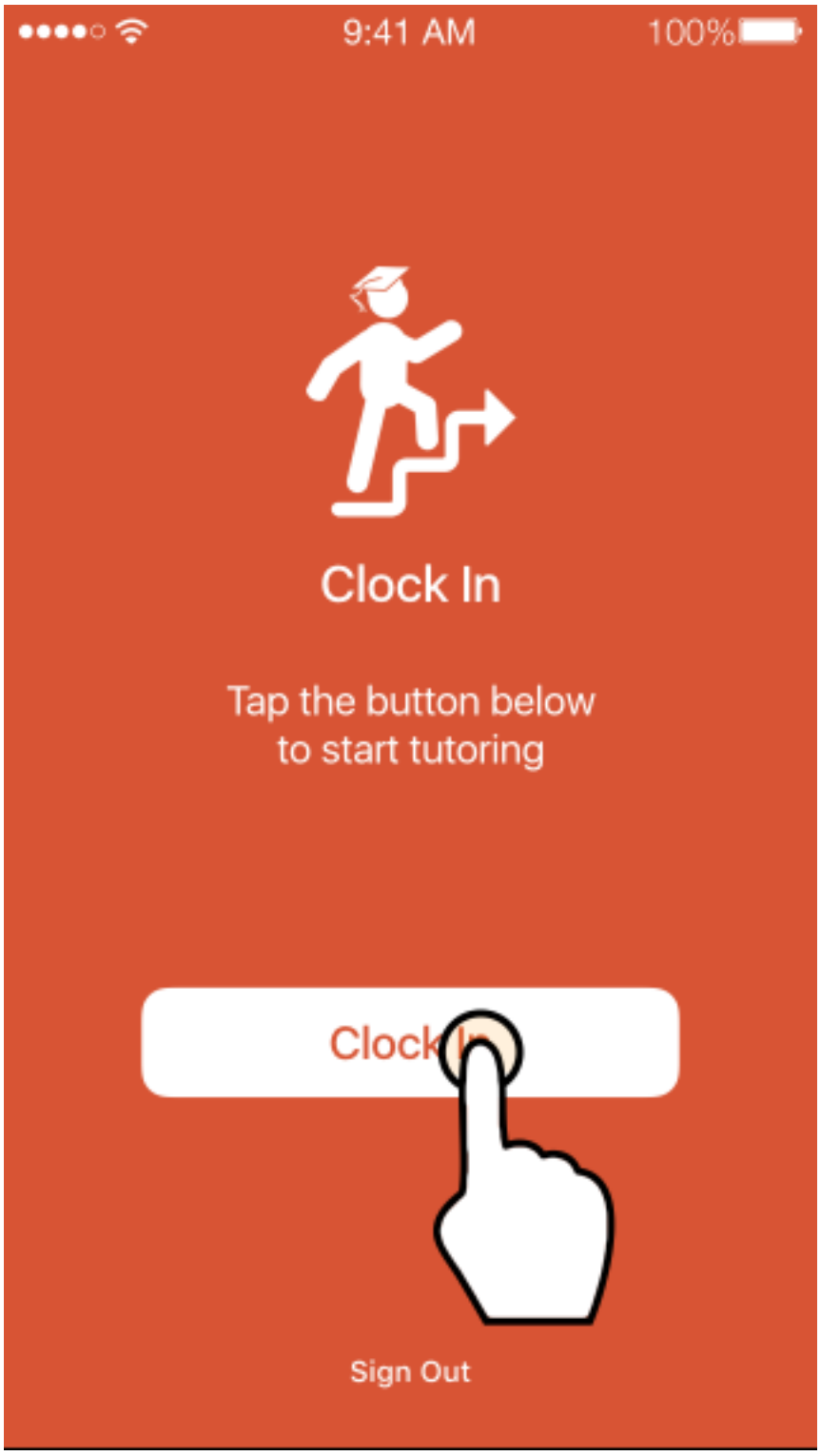
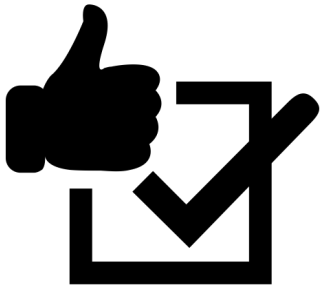


Figure 7: Clock on manually



Chapter 8

API Coding Standards

Contents

1 Doubtfire Git Workflow

We follow a Forking workflow[1] when developing Doubtfire.

1.1 Table of Contents

1.2 About the Doubtfire Branch Structure

We try to keep two main branches at all times:

- `master` for production
- `develop` for current development, a branch off `master`

That way, we follow the workflow:

1. branch off `develop`, giving your branch one of the prefixes defined below,
2. make your changes in that branch,
3. merge your branch back into `develop`,
4. delete your branch to clean up

[1] See `https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow`

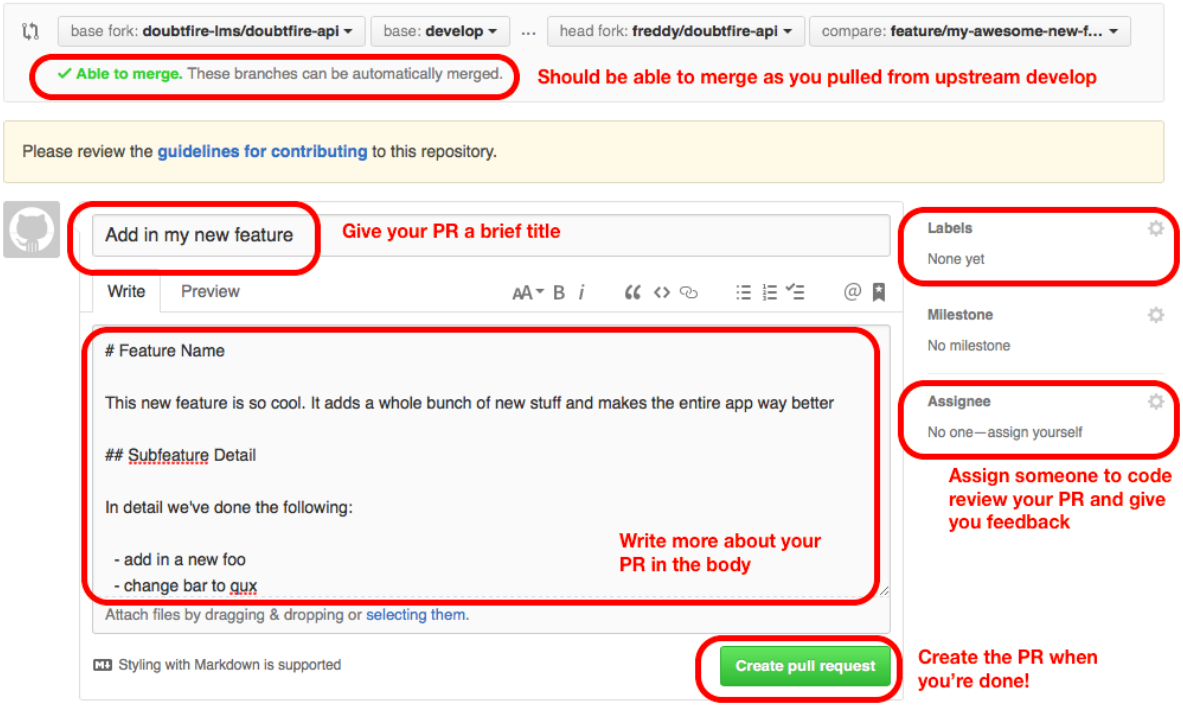


Figure 1: Feature Branches

In some cases, your branches may only consist of one or two commits. This is still okay as you can submit a pull request for code review back into `develop`.

You may want to branch again, e.g.:

```
* master
|\
| \
| |
| (b1) develop
| |\
| | (b2) feature/my-new-feature
| | |\
| | | (b3) test/unit-tests-for-new-feature
| | |/
| | (m1)
| |/
| (m2)
| |\
| | (b4) fix/broken-thing
| |/
| (m3)
| /|
|/ |
(m4)
|  |
|  |
*  *
```

Here, we:

1. branched off `master` to create our `develop` branch, at **`b1`**
2. branched off `develop` to create a new feature under the new branch `feature/my-new-feature`, at **`b2`**
3. branched off `feature/my-new-feature` to create some unit tests for that feature under `test/unit-tests-for-new-feature`, at **`b3`**

4. merged those unit tests back into `feature/my-new-feature`, at **`m1`**
5. merged the new feature back into `develop`, at **`m2`**
6. found a new bug in the feature later on, so branched off `develop` into `fix/broken-thing`, at **`b4`**
7. after we fixed our bug, we merged `fix/broken-thing` back into `develop`, at **`m3`**
8. decide we're ready to release, so merge `develop` into `master`, at **`m4`**

Note that along the way **we're deleting branches after we don't need them**. This helps us keep *short-lived* branches that don't go *stale* after months of inactivity, and prevents us from forgetting about open branches. The only branch we kept open was `develop`, which we can always branch off for new, un-released changes again.

Ideally, any changes that are merged into `master` have been **code-reviewed** before they were merged into `develop`. **You should always code review before merging back into `develop`**. You can do this by performing a Pull Request, where the reviewer can see the changes you want to merge in to `develop`.

1.3 Getting started with the Forking Workflow

1.3.1 1. Forking and Cloning the repository

1.3.1.1 Fork the Repo

To get a copy of a Doubtfire repositories on your user account, you will need to fork it *for each repository*:

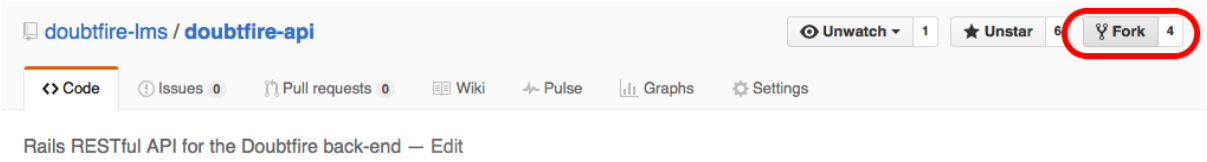


Figure 2: Fork the repo

1.3.1.2 Clone the Fork

You can then clone the repositories you have forked to your machine. To do so, navigate to your forked repositories and copy the clone URL:

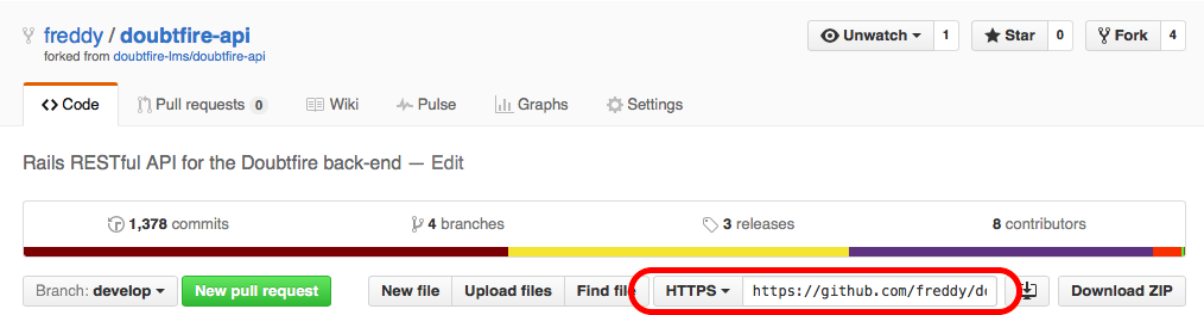


Figure 3: Copy the clone URL

Navigate to your `projects` or `repo` folder, and make a `doubtfire` folder. Then clone using the URLs you copied above:

```
$ cd ~/repos
$ mkdir doubtfire
$ cd doubtfire
$ git clone https://github.com/{username}/doubtfire-api.git
$ git clone https://github.com/{username}/doubtfire-web.git
```

1.3.1.3 Set up your upstream to `doubtfire-lms`

By default, git tracks your remote forked repository (the repository you cloned). This remote is called `origin`.

You will then need to set up a new remote to track to the `doubfire-lms` owned repository. This will be useful when you need to get the latest changes other developers have contributed to the `doubtfire-lms` repo, but you do not yet have those changes in your forked repo. Call this remote `upstream`:

```
$ cd ~/repos/doubtfire/doubtfire-api
$ git remote add upstream https://github.com/doubtfire-lms/doubtfire-api.git
$ cd ~/repos/doubtfire/doubtfire-web
$ git remote add upstream https://github.com/doubtfire-lms/doubtfire-web.git
```

1.3.1.4 Ensure you have your author credentials set up

You should ensure your git user config are set and set to the email address you use with GitHub:

```
$ git config --global user.email "my-github-email@gmail.com"
$ git config --global user.name "Freddy Smith"
```

1.3.1.5 Use a rebase pull

We also want to avoid having merge commits whenever you pull from `upstream`. It is useful to pull from upstream using the `--rebase` switch, as this avoids an unnecessary merge commit when pulling if there are conflicts.

To fix this, always pull with `--rebase` (unless otherwise specified—see the `--ff` switch needed in Step 3):

```
$ git pull upstream develop --rebase
```

or alternatively, make a rebase pull as your default setting:

```
$ git config --global pull.rebase true
```

1.3.2 2. Writing your new changes

As per the branching structure, you need to branch off of `develop` to a new branch that will have your code changes in it. When branching, **be sure you are using a branch prefix**:

```
$ cd ~/repos/doubtfire/doubtfire-api
$ git checkout -b feature/my-awesome-new-feature
```

You can now begin making your changes. Commit along the way, **being sure to conform to the commit message guidelines**, on this branch and push to your fork:

```
$ git status

On branch feature/my-awesome-new-feature
Your branch is up-to-date with 'origin/feature/my-awesome-new-feature'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)
```

```
  modified:   src/file-that-changed.js
  modified:   src/another-file-that-changed.js

$ git add src/file-that-changed.js src/another-file-that-changed.js
$ git commit

[feature/my-awesome-new-feature 7f35016] DOCS: Add new documentation about git
 2 files changed, 10 insertions(+), 15 deletions(-)

$ git push -u origin feature/my-awesome-new-feature
```

Note you only need to add the `-u` flag on an initial commit for a new branch.

1.3.3 3. Prepare for a Pull Request

Note, while it is advised you perform this step, it you can skip it and move straight to the Pull Request step. If the branch cannot be automatically merged, then you should run through these steps.

When you are done with your changes, you need to pull any changes from `develop` from the `upstream` repository. This essentially means "get me anything that has changed on the `doubtfire-lms` repository that I don't yet have".

To do this, pull any changes (if any) from the `upstream` repository's `develop` branch into your local `develop` branch:

```
$ git checkout feature/my-awesome-new-feature
$ git pull --ff upstream develop
```

If there are merge conflicts, you can resolve them now. Follow GitHub's guide[2] for resolving merge conflicts.

We can now update your `origin` repository's `my-awesome-new-feature` on GitHub such that it will include the changes from `upstream`:

```
$ git push origin feature/my-awesome-new-feature
```

[2] See `https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line`

1.3.4 4. Submitting a Pull Request (PR) to the upstream repository

Once you have pushed your changes to your fork, and have ensured nothing has broken, you can then submit a pull request for code review to Doubtfire.

To submit a pull request, go to the relevant Doubtfire LMS Repo and click "New Pull Request":

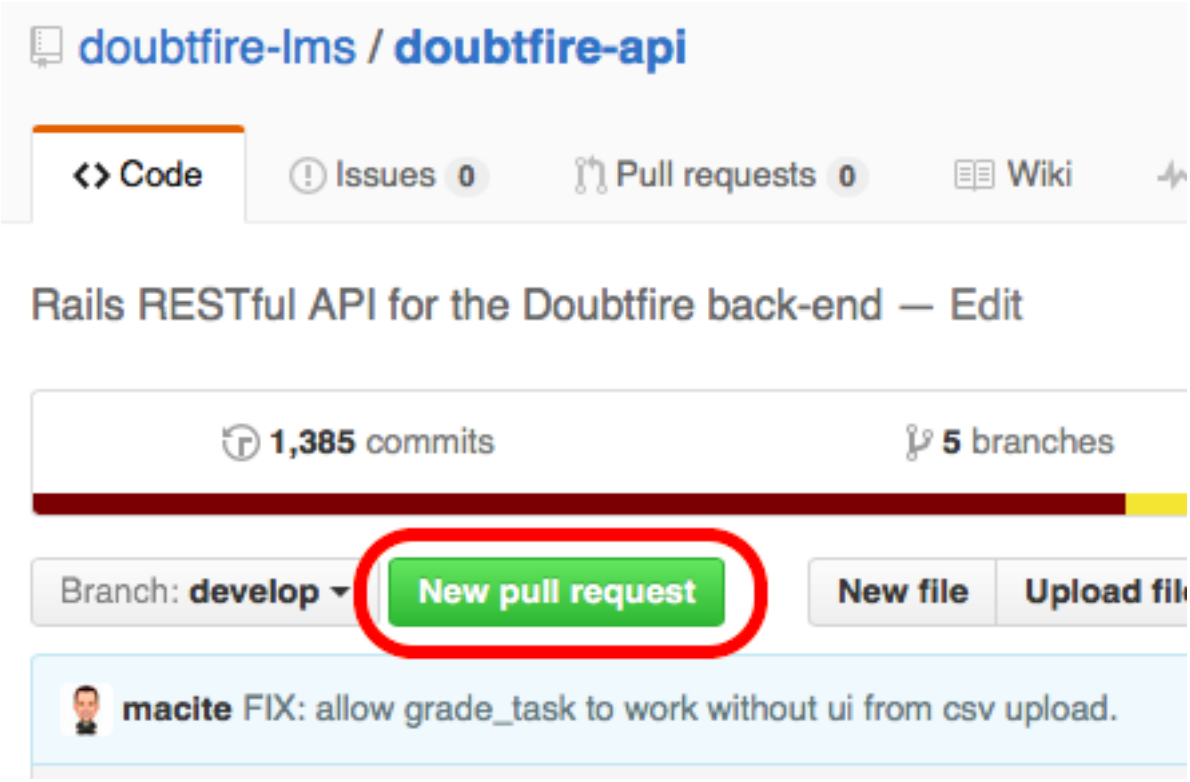


Figure 4: New PR

Ensure that the **Head Fork** is set to your forked repository and on your feature branch. If you cannot see your repository, try clicking the "Compare across forks" link.

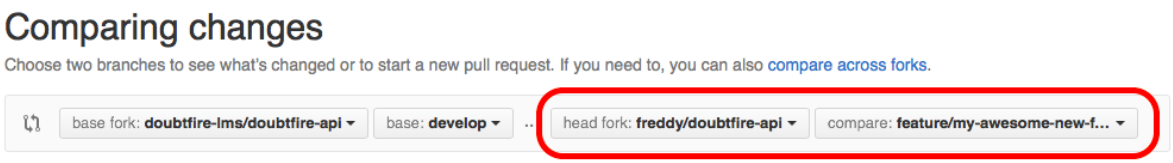


Figure 5: Compare forks

You can then begin writing the pull request. Be sure you are **Able to Merge**, otherwise **try repeating an upstream pull of develop into your feature branch, as per the previous step**.

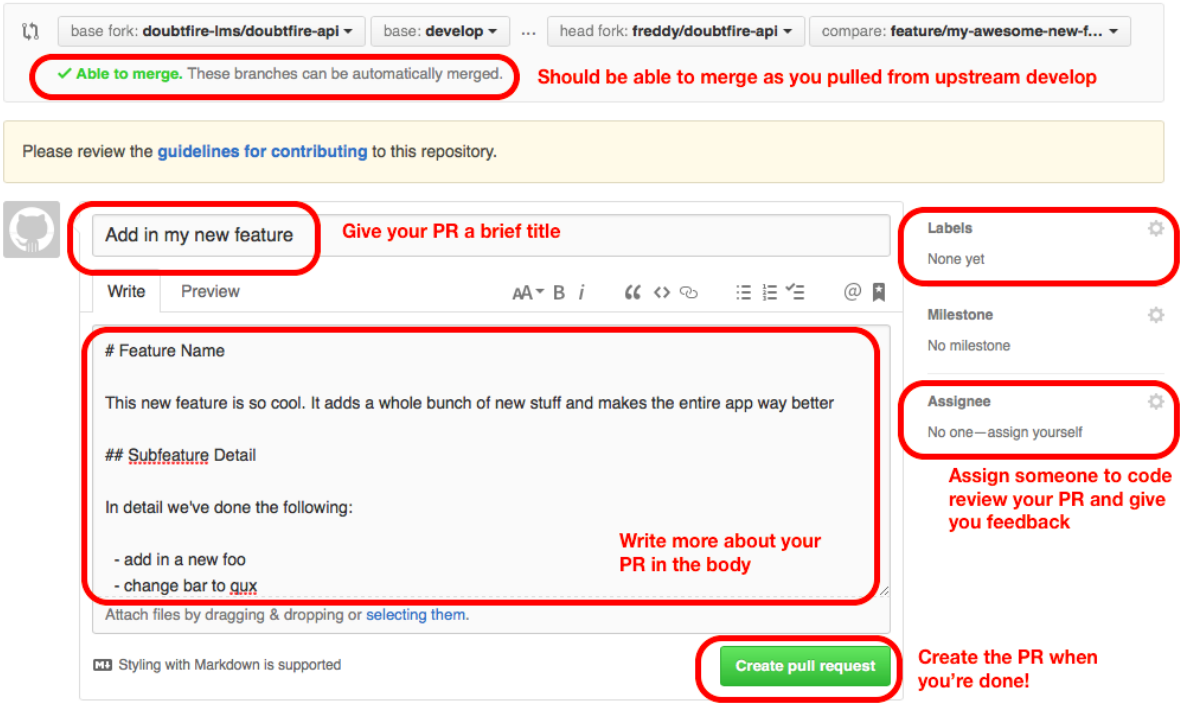


Figure 6: Writing a Pull Request

With your PR body, be descriptive. GitHub may automatically add a commit summary in the body. If fixing a problem, include a description of the problem you're trying to fix and why this PR fixes it. When you are done, assign a code reviewer and add a tag (if applicable) and create the pull request!

If your code is ok, it will be merged into `develop`, (and eventually `master`, meaning your code will go live - woohoo :tada:)

If not, the reviewer will give you suggestions and feedback for you to fix your code.

STOP! Continue to the next step once your Pull Request is approved and merged into the `doubtfire-lms`'s `develop` branch.

1.3.5 5. Cleaning Up

Once your pull request is approved, your code changes are finalised, and merged you will want to delete your old feature branch so you don't get lots of old branches on your repository.

Following from the example above, we would delete `feature/my-awesome-new-feature` as it has been merged into `develop`. We first delete the branch locally:

```
$ git branch -D feature/my-awesome-new-feature
```

Then remove it from your fork on GitHub:

```
$ git push origin --delete feature/my-awesome-new-feature
```

Then ensure you are git is no longer tracking the deleted branch from `origin` by running a fetch prune:

```
$ git fetch --prune
```

As your changes have been merged into `upstream`'s `develop` branch, pull from `upstream` and you can grab those changes into your local repository:

```
$ git checkout develop
$ git pull upstream develop
```

Then push those changes up into your `origin`'s `develop` so that it is synced with `upstream`'s `develop`:

```
$ git push origin upstream
```

1.3.6 Workflow Summary

Step 1. Set up for new feature branch:

```
$ git checkout develop                  # make sure you are on develop
$ git pull --rebase upstream develop    # sync your local develop with upstream's develop
$ git checkout -b my-new-branch         # create your new feature branch
```

Step 2. Make changes, and repeat until you are done:

```
$ git add ... ; git commit ; git push     # make changes, commit, and push to origin
```

Step 3. Submit a pull request, and **if unable to merge**:

```
$ git pull --ff upstream develop          # merge upstream's develop in your feature branch
$ git add ... ; git commit                # resolve merge conflicts and commit
$ git push origin                         # push your merge conflict resolution to origin
```

Step 4. Only when the pull request has been **approved and merged**, clean up:

```
$ git checkout develop                    # make sure you are back on develop
$ git branch -D my-new-branch             # delete the feature branch locally
$ git push --delete my-new-branch         # delete the feature branch on origin
$ git fetch origin --prune                # make sure you no longer track the deleted branch
$ git pull --rebase upstream develop      # pull the merged changes from develop
$ git push origin develop                 # push to origin to sync origin with develop
```

1.4 Branch Prefixes

When branching, try to prefix your branch with one of the following prefixes shown in Table 1.

1.5 Writing Commit Messages

Parts of this section have been adapted from Chris Beam's post, How to Write Good Commit Messages[3].

When writing commits, try to follow this guide as described in this subsection.

1.5.1 Prefix your commit subject line with a tag

Each one of your commit messages should be prefixed with one of the following shown in Table 2

1.5.2 Formatting your message

Capitalise your commit messages and do not end the subject line with a period

```
FIX: Change the behaviour of the logging system
```

[3]See `http://chris.beams.io/posts/git-commit/`

Table 1: Branch prefixes

Prefix	Description	Example
`feature/`	New feature was added	`feature/add-learning-outcome-alignment`
`fix/`	A bug was fixed	`fix/crash-when-code-submission-finished`
`enhance/`	Improvement to existing feature, but not visual enhancement (See `LOOKS`)	`enhance/allow-code-files-to-be-submitted`
`looks/`	UI Refinement, but not functional change (See `ENHANCE`)	`looks/rebrand-ui-for-version-2-marketing`
`quality/`	Refactoring of existing code	`quality/make-code-convention-consistent`
`doc/`	Documentation-related changes	`doc/add-new-api-documentation`
`config/`	Project configuration changes	`config/add-framework-x-to-project`
`speed/`	Performance-related improvements	`speed/new-algorithm-to-process-foo`
`test/`	Test addition or enhancement	`test/unit-tests-for-new-feature-x`

Table 2: Commit tagging guide

Tag	Description	Example
`NEW`	New feature was added	**NEW**: Add unit outcome alignment tab
`FIX`	A bug was fixed	**FIX**: Amend typo throwing error
`ENHANCE`	Improvement to existing feature, but not visual enhancement (See `LOOKS`)	**ENHANCE**: Calculate time between classes to show on timetable
`LOOKS`	UI Refinement, but not functional change (See `ENHANCE`)	**LOOKS**: Make plagiarism tab consistent with other tabs
`QUALITY`	Refactoring of existing code	**QUALITY**: Make directives in consistent format with eachother
`DOC`	Documentation-related changes	**DOC**: Write guide on writing commit messages
`CONFIG`	Project configuration changes	**CONFIG**: Add new scheme for UI automation testing
`SPEED`	Performance-related improvements	**SPEED**: Reduce time needed to batch process PDF submissions
`TEST`	Test addition or enhancement	**TEST**: Add unit tests for tutorial administration

and not

```
fix: change the behaviour of the logging system.
```

1.5.3 Use the imperative mood in your commit subject line

Write your commits in the imperative mood and not the indicative mood

- "Fix a bug" and **not** "Fix*ed* a bug"
- "Change the behaviour of Y" and **not** "*Changed* the behaviour of Y"
- "Add new API methods" and **not** "Sweet new API methods"

A properly formed git commit subject line should always be able to complete the following sentence:

> If applied, this commit will **your subject line here**
>
> If applied, this commit will **fix a bug**
>
> If applied, this commit will **change the behaviour of Y**

and not

> If applied, this commit will **sweet new API methods**

1.5.4 Subject and body lines

Write a commit subject, and explain that commit on a new line (if need be):

```
FIX: Derezz the master control program

MCP turned out to be evil and had become intent on world domination.
This commit throws Tron's disc into MCP (causing its deresolution)
and turns it back into a chess game.
```

Keep the subject line (top line) concise; keep it **within 50 characters**.

Use the body (lines after the top line) to explain why and what and *not* how; keep it **within 72 characters**.

1.5.4.1 But how can I write new lines if I'm using `git commit -m "Message"`?

Don't use the `-m` switch. Use a text editor to write your commit message instead.

If you are using the command line to write your commits, it is useful to set your git editor to make writing a commit body easier. You can use the following command to set your editor to `nano`, `emacs`, `vim`, `atom`.

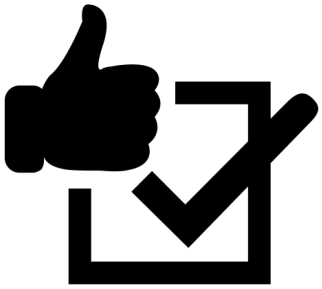
```
$ git config --global core.editor nano
$ git config --global core.editor emacs
$ git config --global core.editor vim
$ git config --global core.editor "atom --wait"
```

If you want to use Sublime Text as your editor, follow this guide[4].

If you are not using the command line for git, you probably should be[5].

[4] See `https://help.github.com/articles/associating-text-editors-with-git/#using-sublime-text-as-your-editor`

[5] See `http://try.github.io`



Chapter 9

Web Coding Standards

Contents

1 Contributing

Please read through this document before contributing to Doubtfire.

Before continuing, **please read the contributing document[1] of the API**, as this outlines the Git workflow you should be following.

1.1 Coding Guidelines

For extendability and maintenance purposes, following these guidelines:

- Name a directive with it's role in mind (i.e., as a **Agent Noun**[2]) to give a small summary as to what the directive *does*:
- when *viewing* a project or task, the directive is project-**viewer** and task-**viewer**
- when *assessing* task submissions, the directive is task-submission-**assessor**
- when *editing* a unit's tutorials, the directive is unit-tutorial-**editor**
- Name directives that show lots of data in one directive in a table a list: e.g.: `unit-student-list`, `group-member-list`, `project-top-task-list`
- Name directives with a series of steps to perform a goal a `wizard`, e.g.: `project-portfolio-wizard`, `new-user-wizard`
- Always name modals in Pascal Case `SomeModal` and create them as a factory/controller pair CoffeeScript file which can then be easily created on the fly:

```
# foo/modals/create-foo-modal.coffee
angular.module('doubfire.foo.modals.create-foo-modal', [])

#
# Prompts the user to create a Foo using a bar and qux variable
#
.factory('CreateFooModal', ($modal) ->
  CreateFooModal = {}

  CreateFooModal.show = (bar, qux) ->
```

[1] See `https://github.com/doubtfire-lms/doubtfire-api/blob/develop/CONTRIBUTING.md`
[2] See `https://en.wikipedia.org/wiki/Agent_noun`

```
    $modal.open
      templateUrl: 'foo/modals/create-foo-modal.tpl.html'
      controller: 'CreateFooModalCtrl'
      resolve:
        bar: -> bar
        qux: -> qux

  CreateFooModal
)

.controller('CreateFooModalCtrl', ($scope, bar, qux) ->
  # Does stuff with bar and qux to create a foo
  $scope.bar = bar
  $scope.qux = qux
)

# foo/states/foo-view/foo-view.coffee
# ...
.controller('FooViewCtrl', ($scope, CreateFooModal) ->
  # ...
  $createNewFoo = ->
    CreateFooModal.show($scope.bar, $scope.qux)
)
```

- Always name non-anonymous controllers with a `Ctrl` suffix
- Case correctly:
- `directiveName` should be camelCase - refer to this Angular documentation[3]
- `ServiceName`, `ControllerNameCtrl` should be in PascalCase
- Regardless of abbreviations, stick to these conventions (e.g., `pdfPanelViewer` directive works, but `PDFPanelViewer` won't work as it needs to be camelCase)
- Place modals and states in a `modals` and `states` folder under the root. All else can be in their own folders unless they are of a related concept (see the `project-portfolio-wizard` folder under `project`, `stats` under `tasks` and `units`)

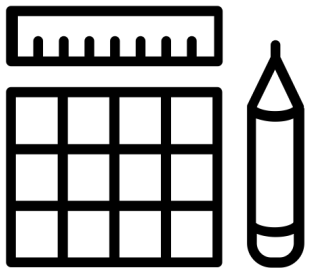
[3] See `https://docs.angularjs.org/guide/directive#normalization`

- The name of a module should follow the directory structure of where it has been placed (i.e., in the above example, the template file was at `foo/modals/create-foo-modal.tpl.html`, the CoffeeScript file was at `foo/modals/create-foo-modal.coffee`, and thus the module is `doubtfire.foo.modals.create`
- Try to give a brief summary of what the directive, state or factory does. E.g., the comment in the example above for `CreateFooModal` is sufficient.
- Try to abstract as much code inside a model class as possible. At present a lot of this code is in a model's service, and it should be moved into the model's resource definition as much as possible:

```
Unit.addTutorial tutorialData
```

instead of:

```
unitService.addTutorial unit, tutorialData
```



Chapter 10

Design Prototype

Contents

List of Figures

1 Invision

Prototypes have been designed using InVision[1]

2 iOS Prototype

The iOS prototype is hosted at `http://invis.io/4S76XFOKB`.

2.1 Tutor Queue

The global queue (Figure 1) is shown when there are pending tickets. It is separated between recommended and all pending students. Recommended students show students who need help with subjects which the tutor teaches. Badges on the side show the task abbreviation which students need help with, if applicable.

Tutors can accept or defer tickets directly from their global queue, as shown in Figure 2.

The empty tutor queue is the queue when the ticket is empty. This means there are no tickets in the tutor's local queue, meaning the tutor is not currently helping any students. Figure 3 shows this, whereas Figure 4 shows a queue with more students being helped by the tutor.

A tutor may choose to postpone a student directly from their queue (Figure 5). If a tutor skips the first student at the top of the list, the student who has been seen the longest time ago or has never been seen at all, then the app warns them. This is shown in the alert under Figure 6.

2.2 Ticket View

Tapping a ticket shows the details for the ticket. The same is shown for both the global and tutor queue, however the primary actions differ. This is shown under Figure 7

When viewing a ticket from the tutor queue (Figure 8), the ticket can be marked as resolved or deferred (i.e., "Come Back Later"). When the ticket is viewed from the global queue, it can be taken aboard by the tutor (accept ticket) or referred to another tutor.

Referring the ticket to another tutor shows tutors who are currently working at the helpdesk and adds their ticket to their queue. This workflow is illustrated under Figure 9.

[1] See `http://invisionapp.com`

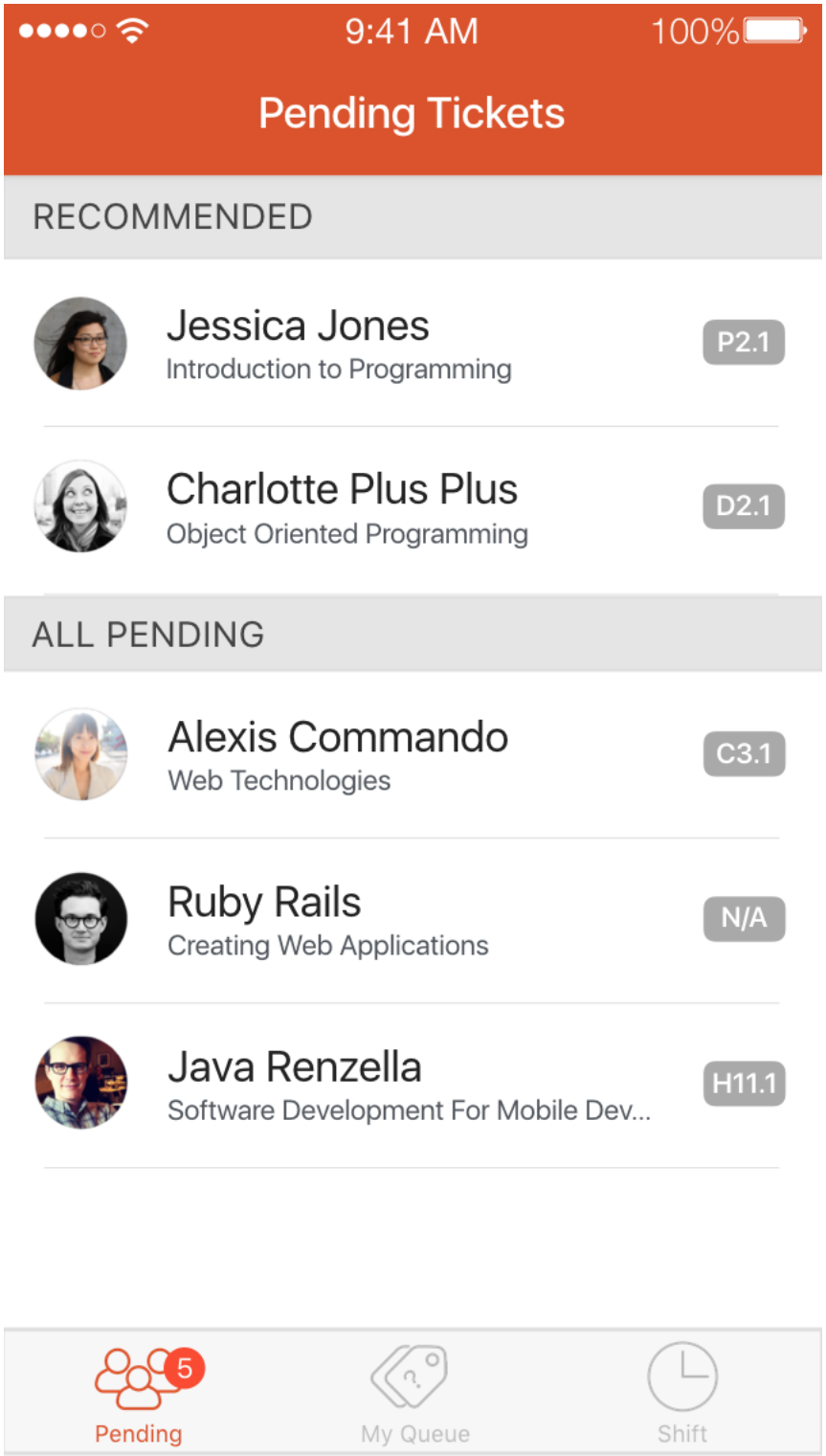


Figure 1: Global Queue

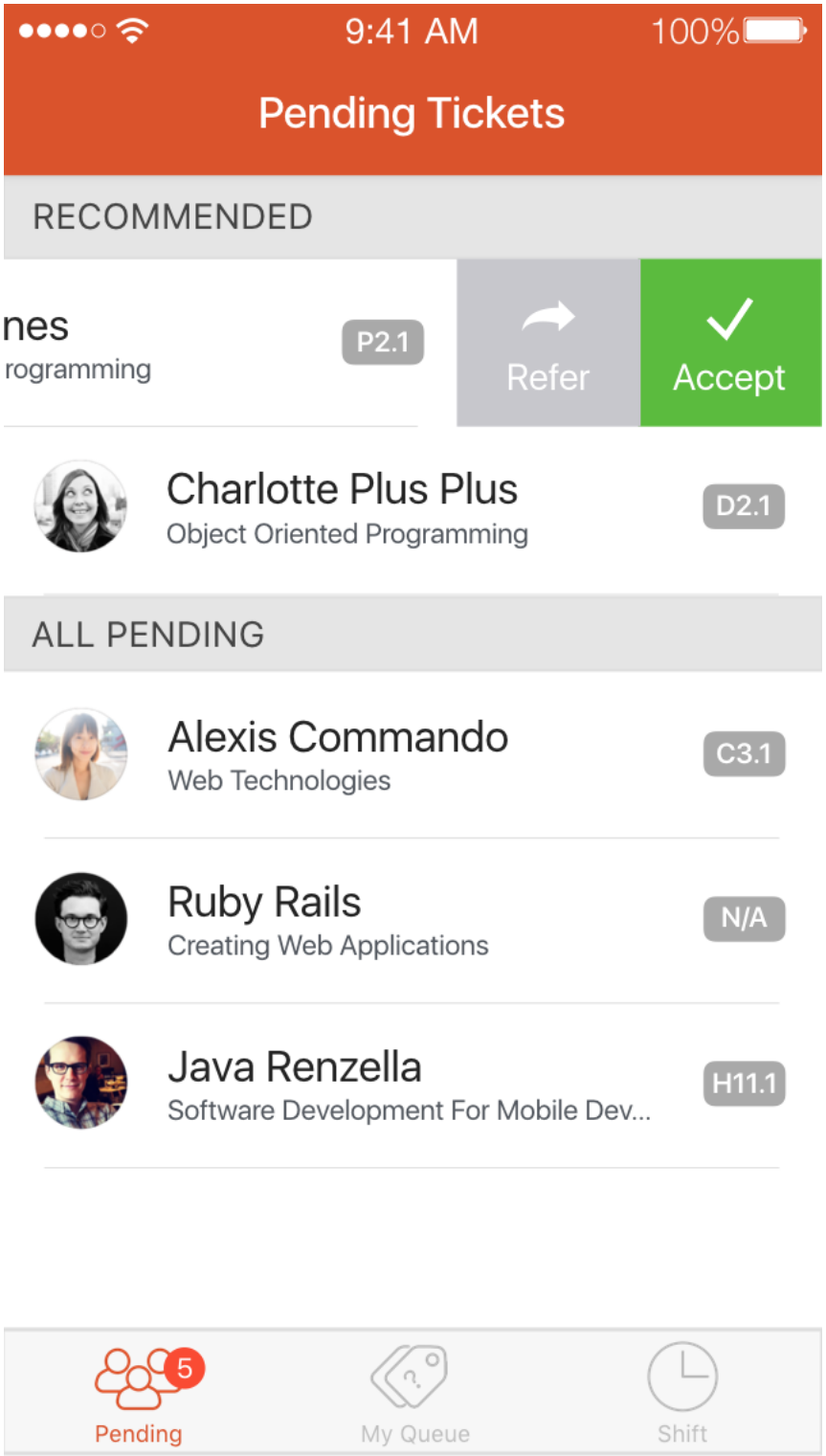


Figure 2: Accept or Defer directly from the queue

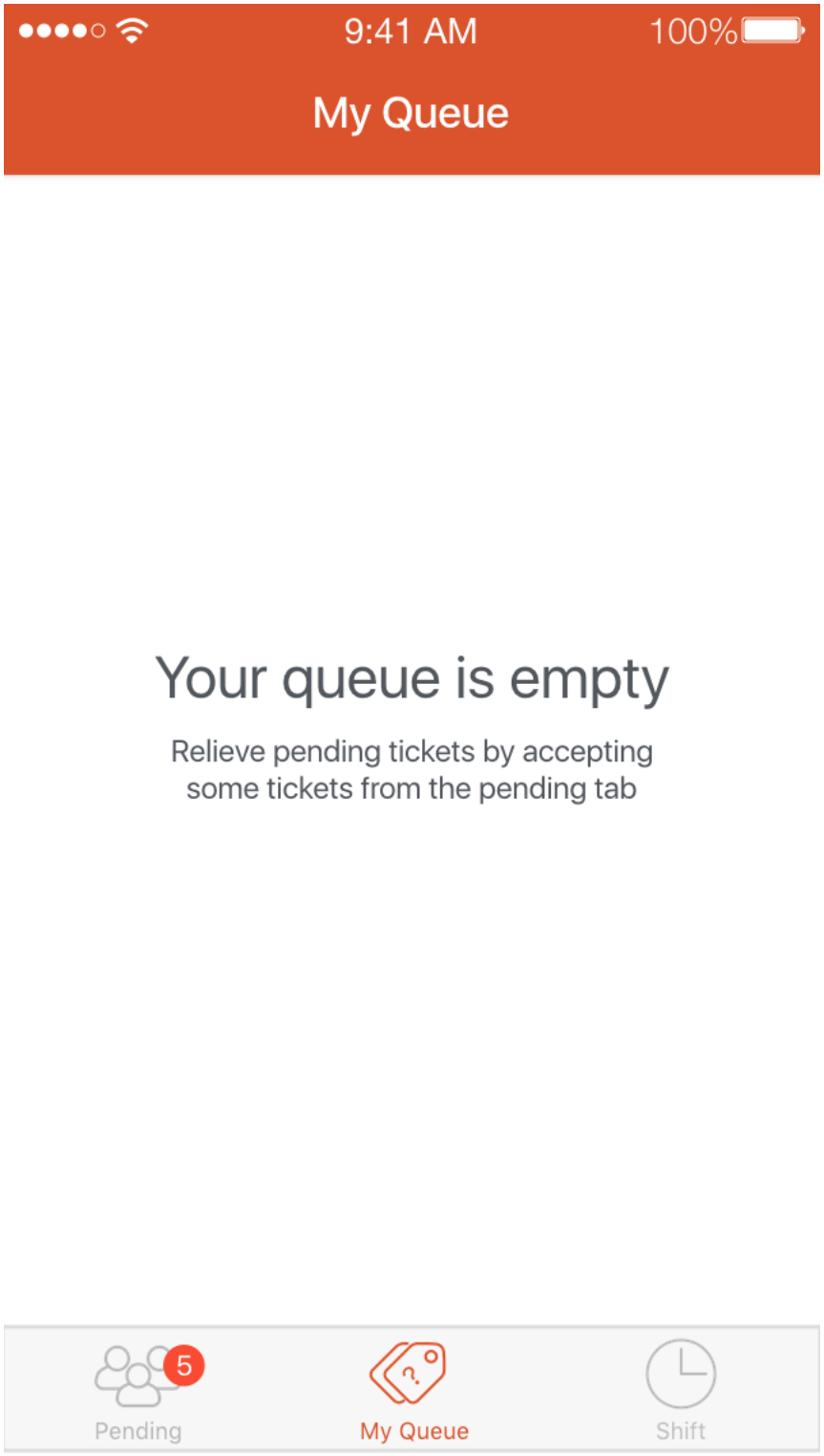


Figure 3: Empty Tutor Queue

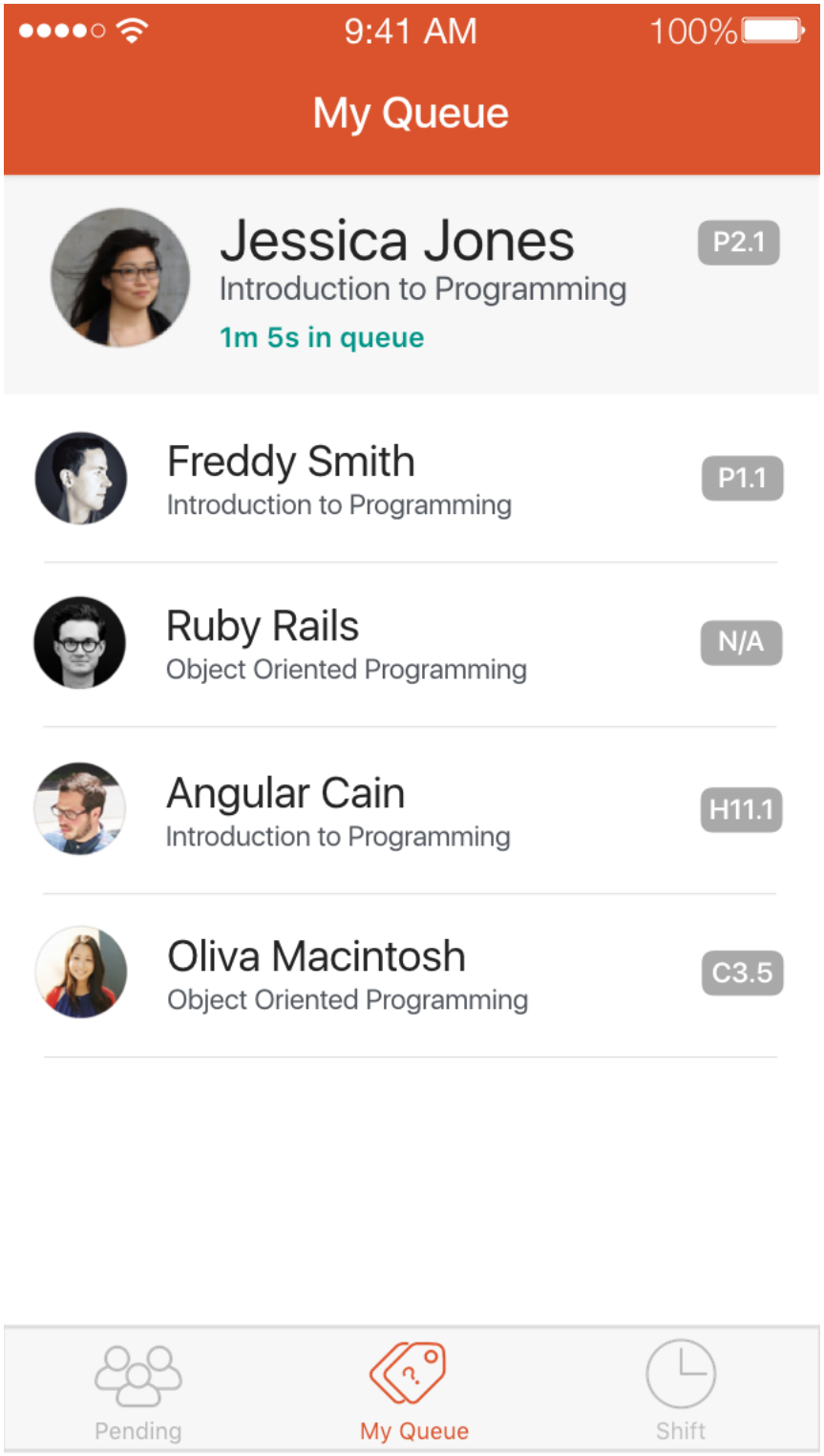


Figure 4: Full Queue

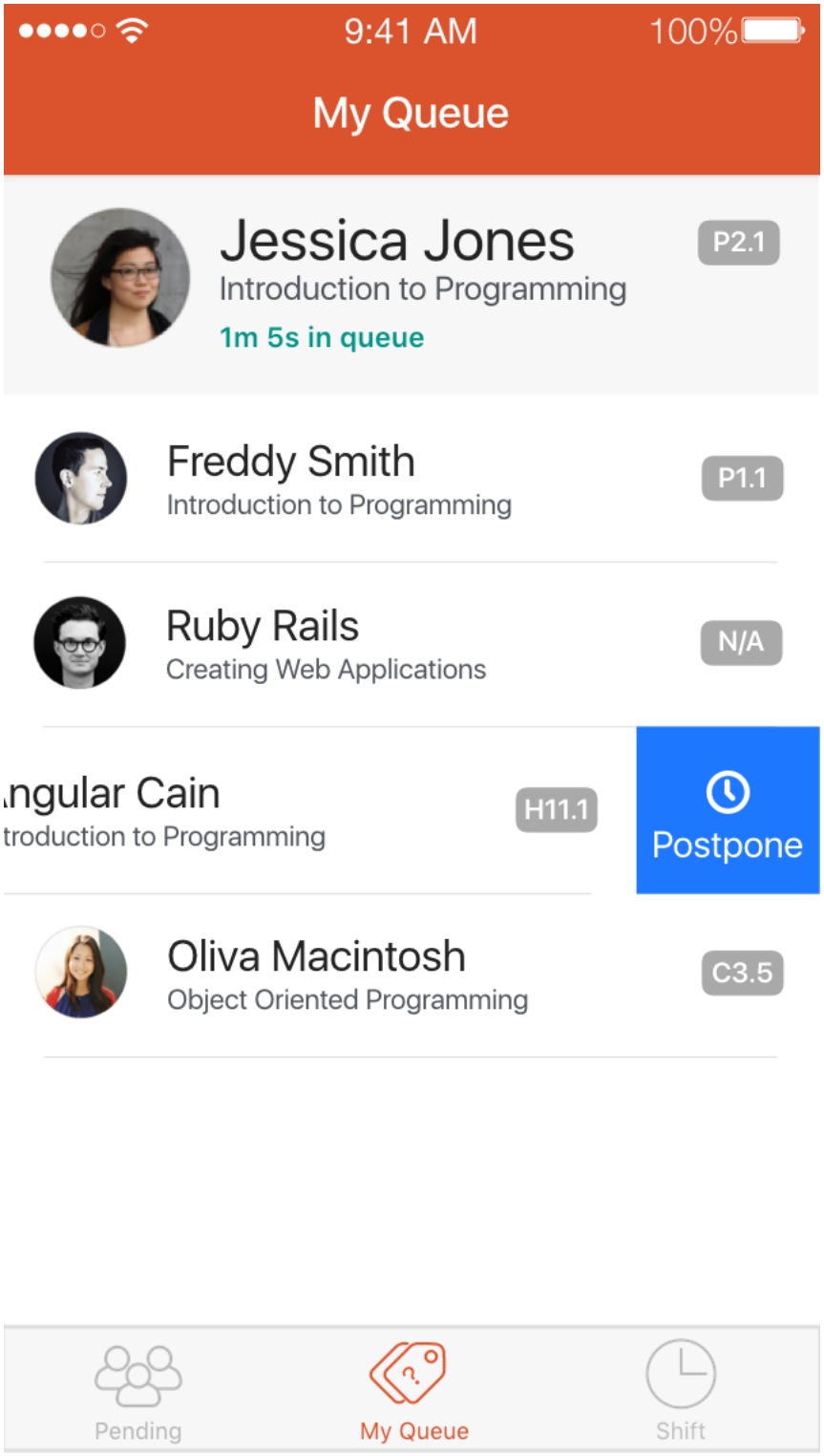


Figure 5: Postpone Student Directly from the Queue

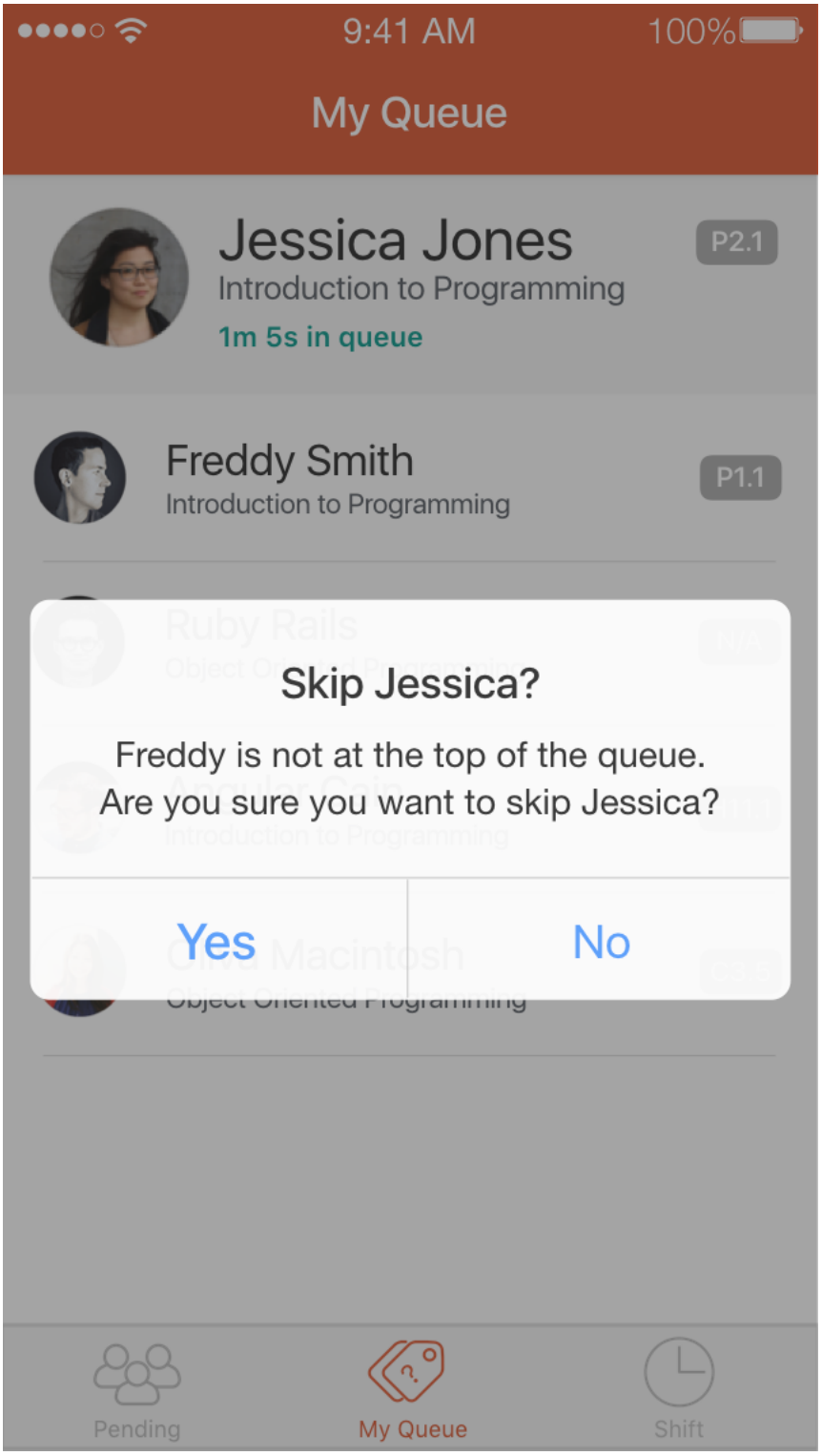


Figure 6: Skip First Student Warning

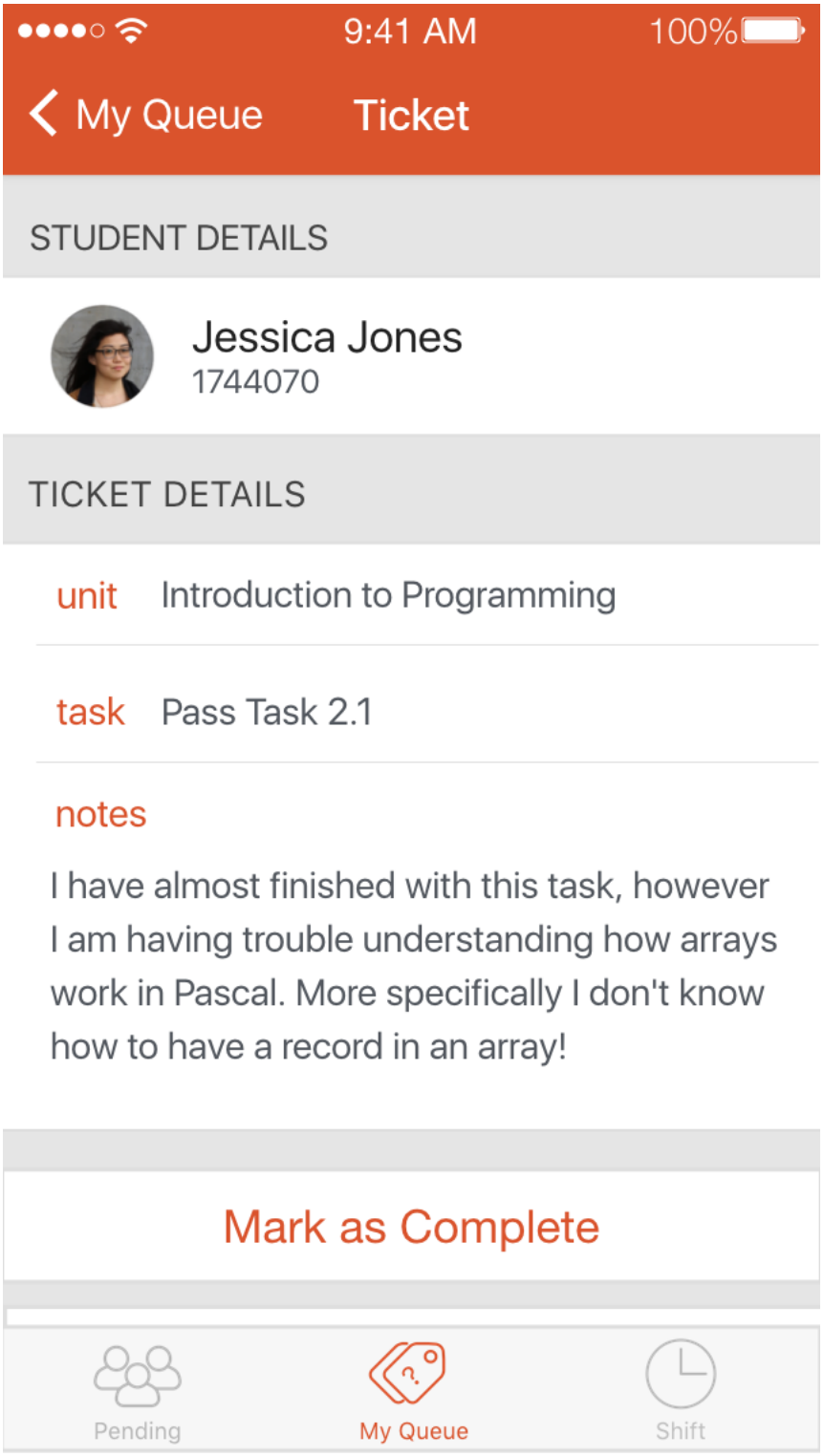


Figure 7: Tapping a student from the tutor queue

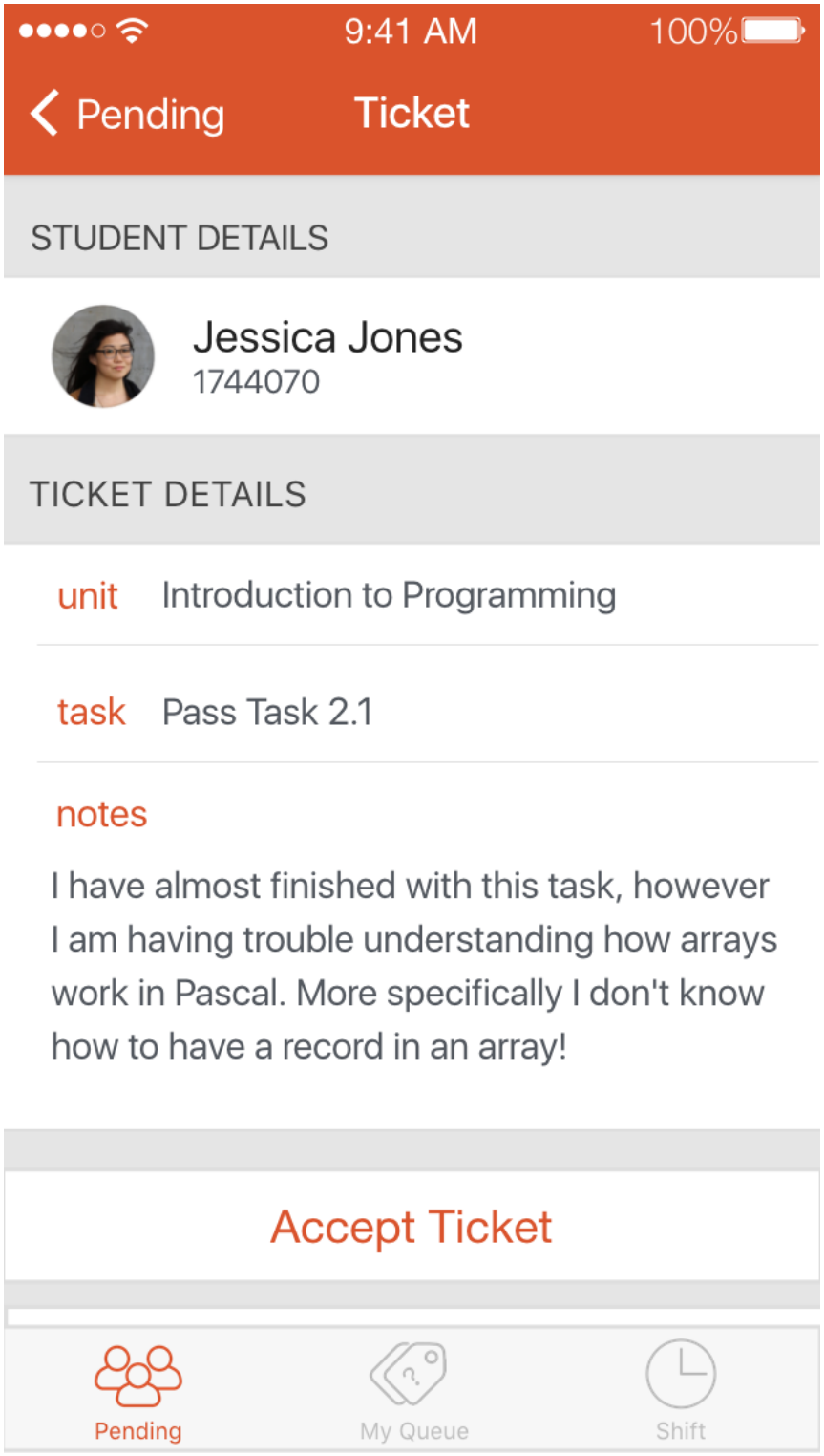


Figure 8: Tapping a Student from the global queue

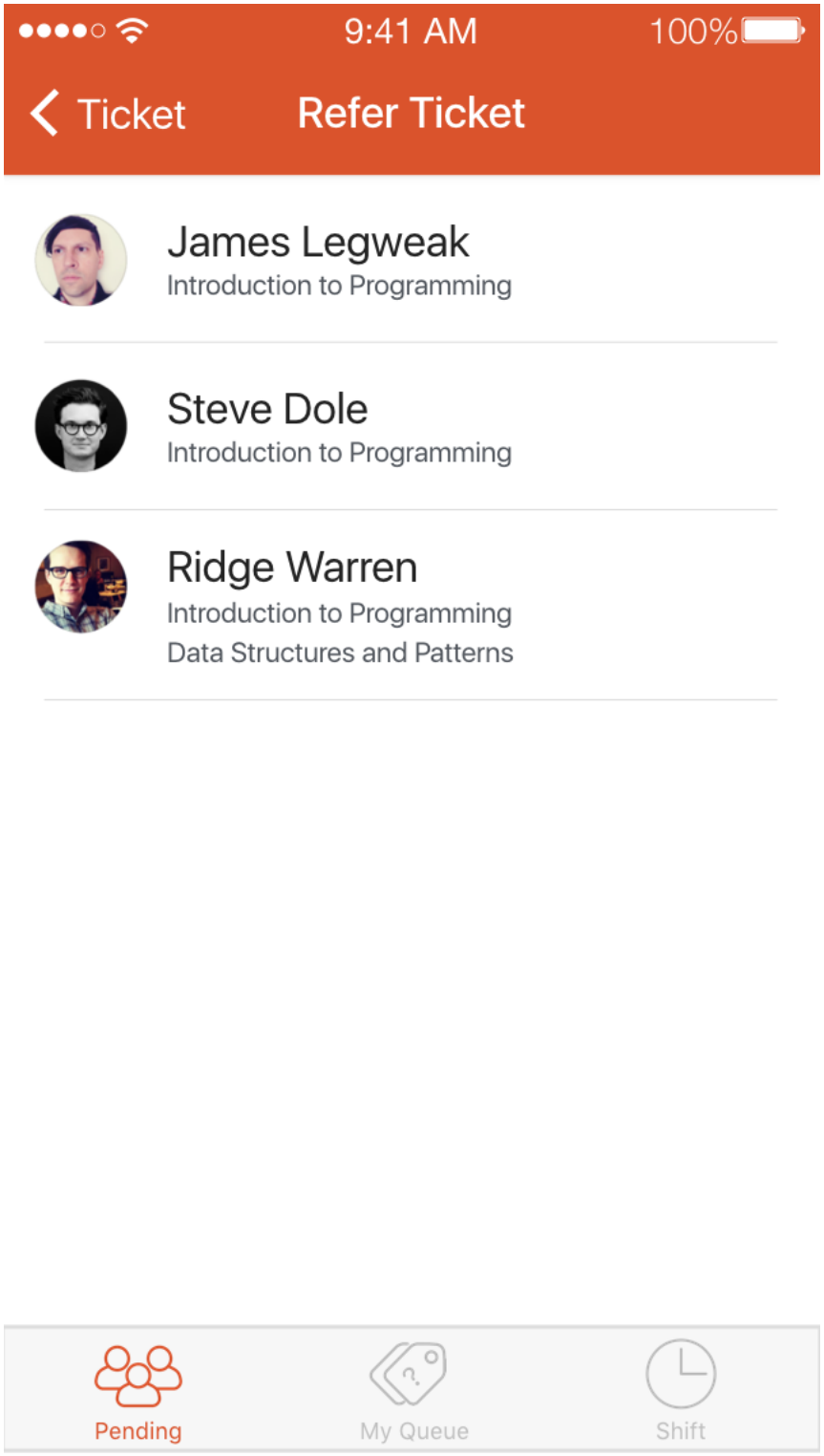


Figure 9: Referring a student to another tutor

2.3 Push Notifications

Approaching the helpdesk connects to the BLE-enabled device at the helpedesk and prompts them to clock on. When leaving the helpdesk, a push notification will warn them that they are out of range and thus have clocked off.

Figures 10 and 11 show this in more detail.

2.4 Miscellaneous

A basic sign in screen will use standard Doubtfire authentication, i.e., SIMS. This is shown under the login view depicted in Figure 12.

The about screen will show basic details, such as the Open-Source license, as illustrated in Figure 13.

When bluetooth is disabled, a warning will be shown, preventing tutors to continue using the app (Figure14).

When a tutor disables auto-clock-on, then they can manually clock on by this screen. This is shown in Figure 15.

The shift tab shows details about the tutor's current shift (Figure 16). From the shift tab, a tutor can manually clock off (Figure 17).

3 Android Prototype

The Android prototype is hosted at `http://invis.io/YF6Z1WRBM`. It follows the same workflow as the iOS app, except with Android-specific user design paradigms.

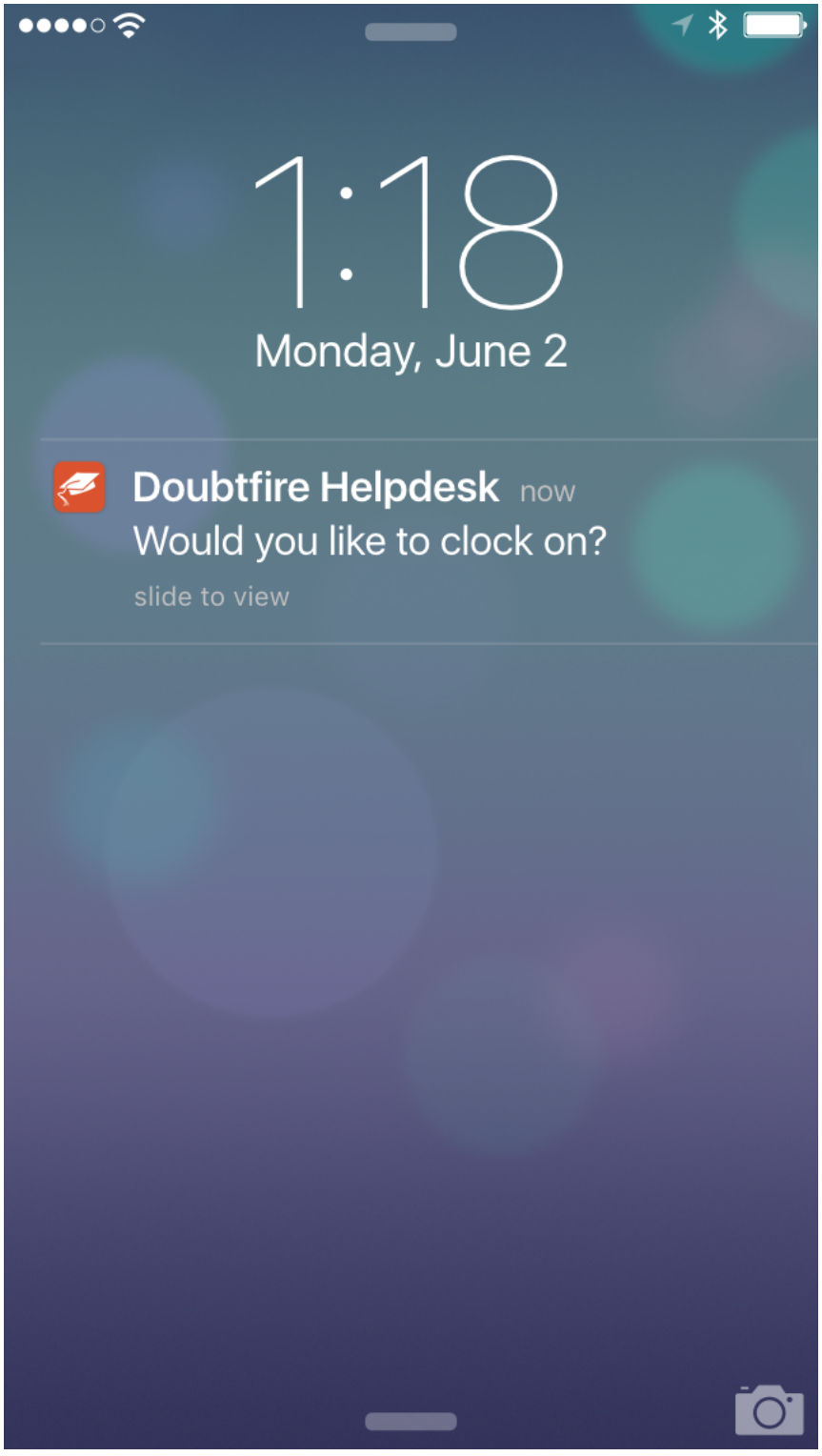


Figure 10: Push Notification - Clock On

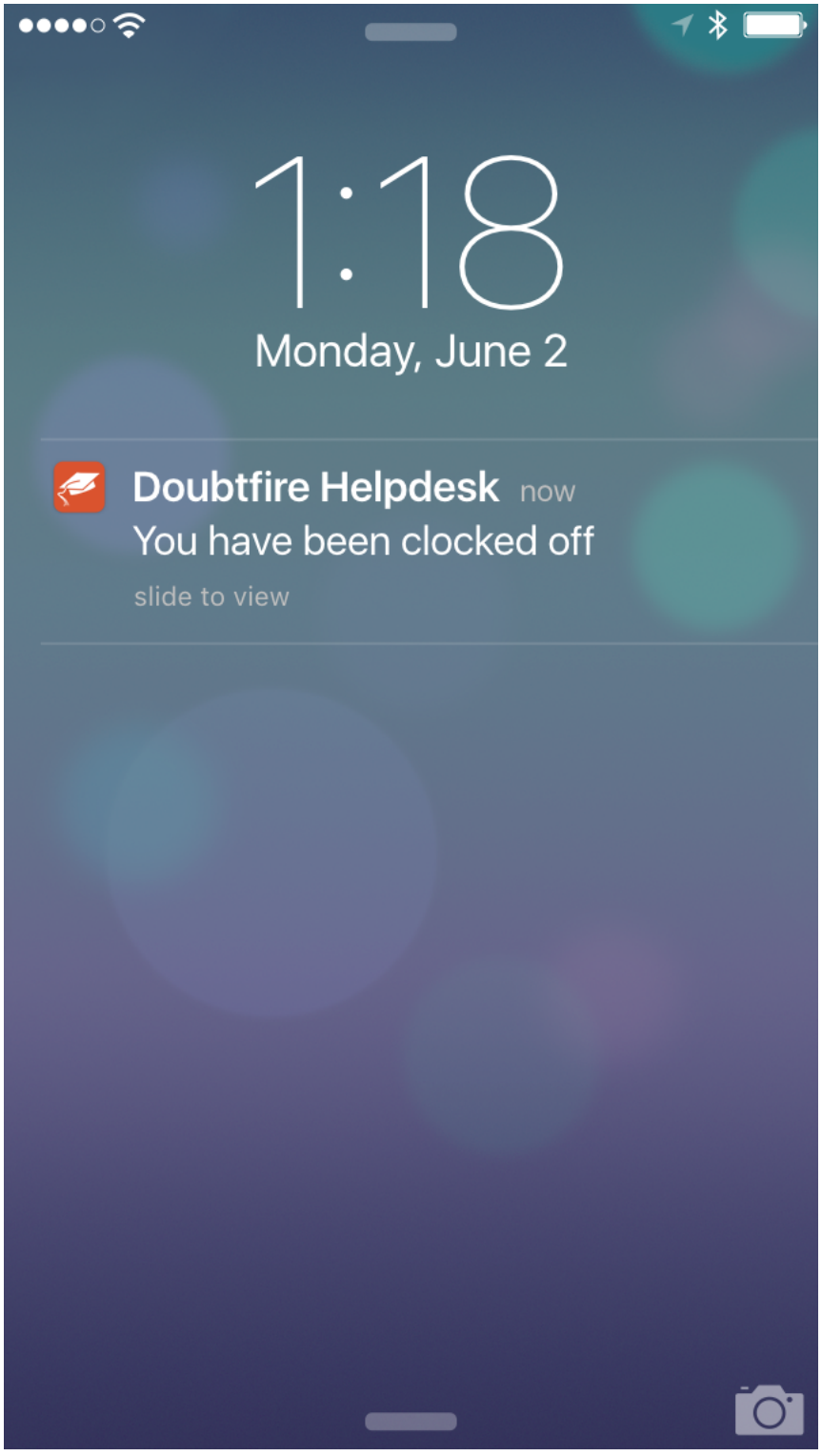


Figure 11: Push Notification - Clock Off

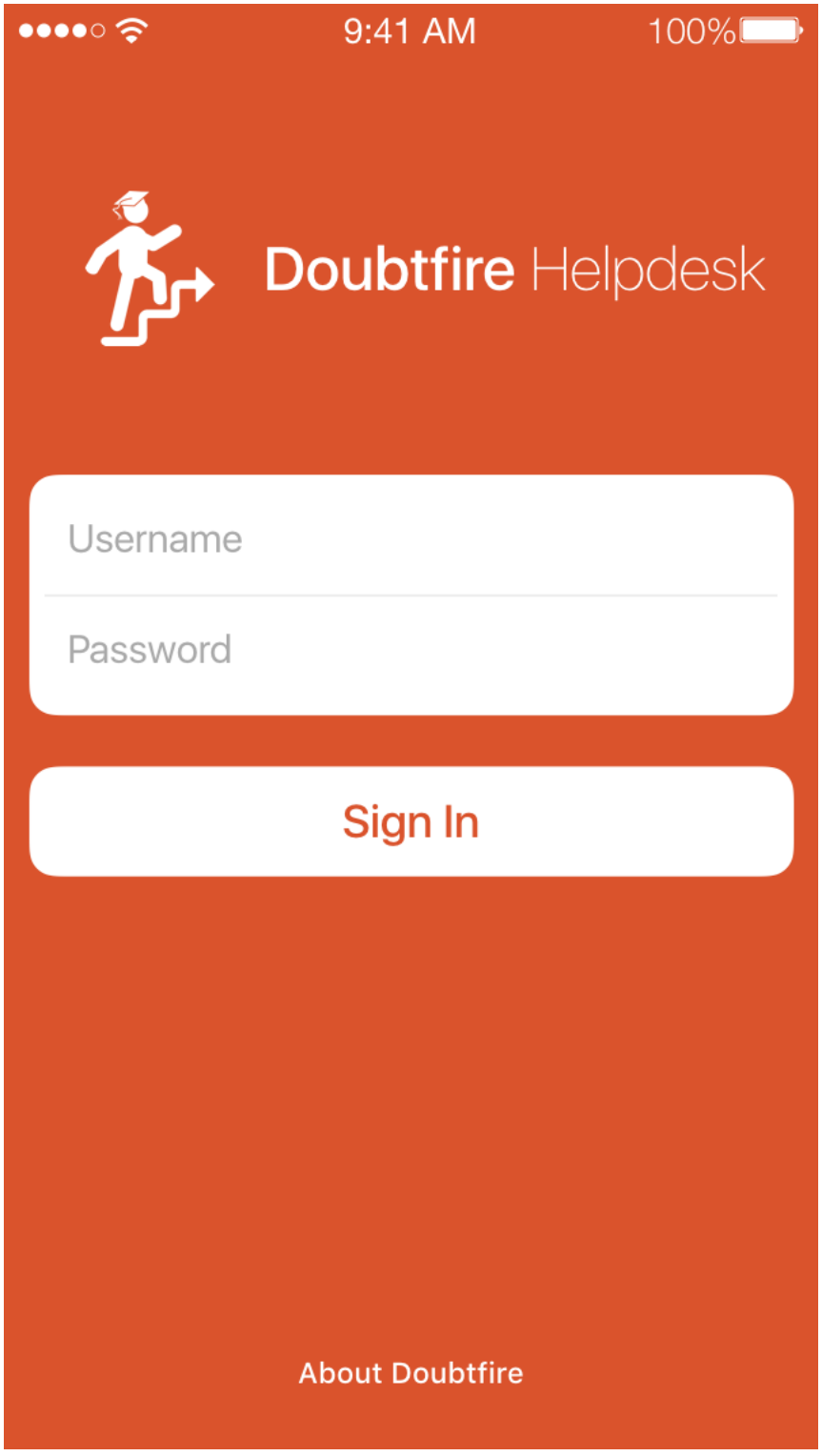


Figure 12: Sign In

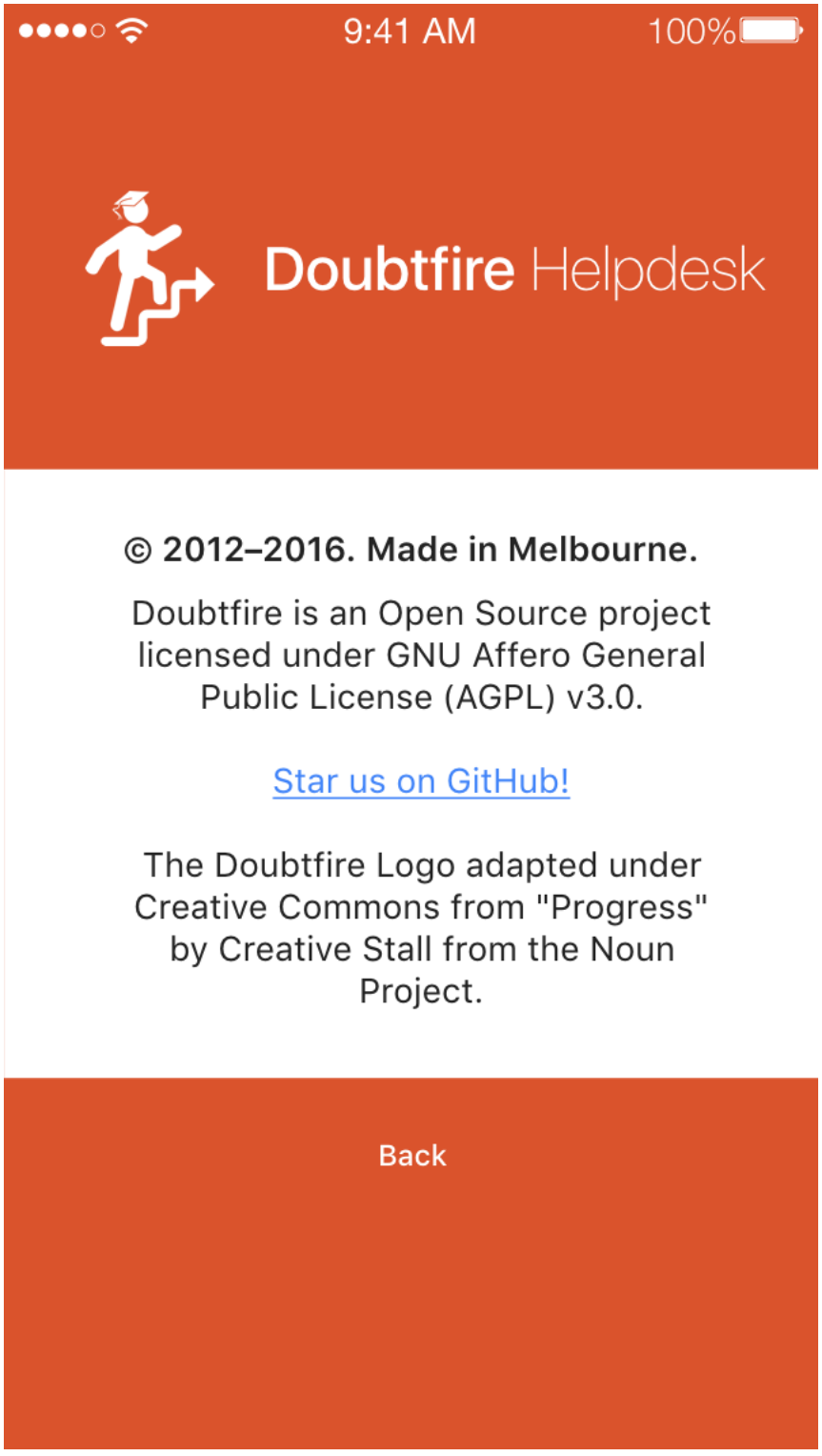


Figure 13: About Doubtfire App

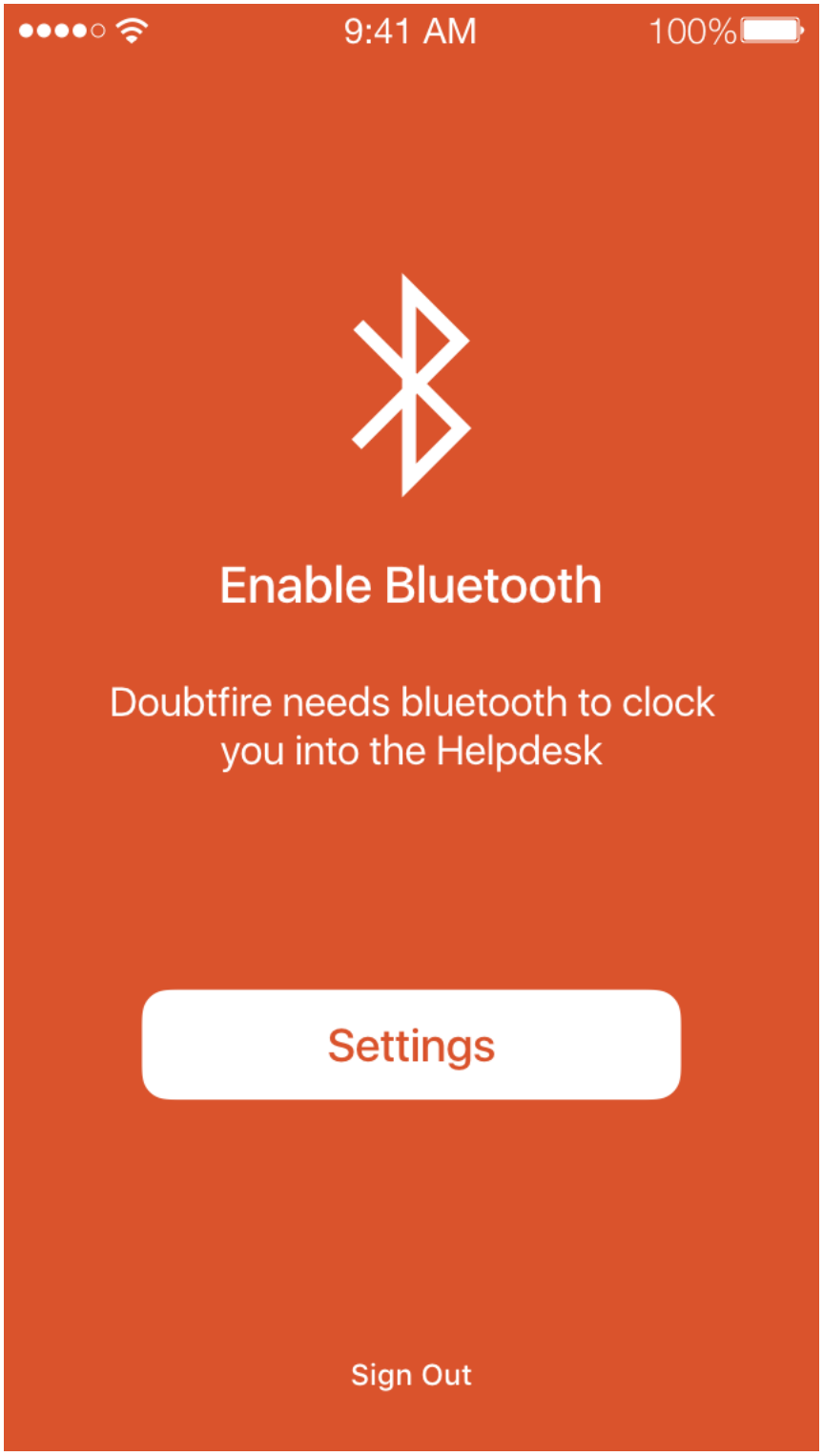


Figure 14: Enable Bluetooth

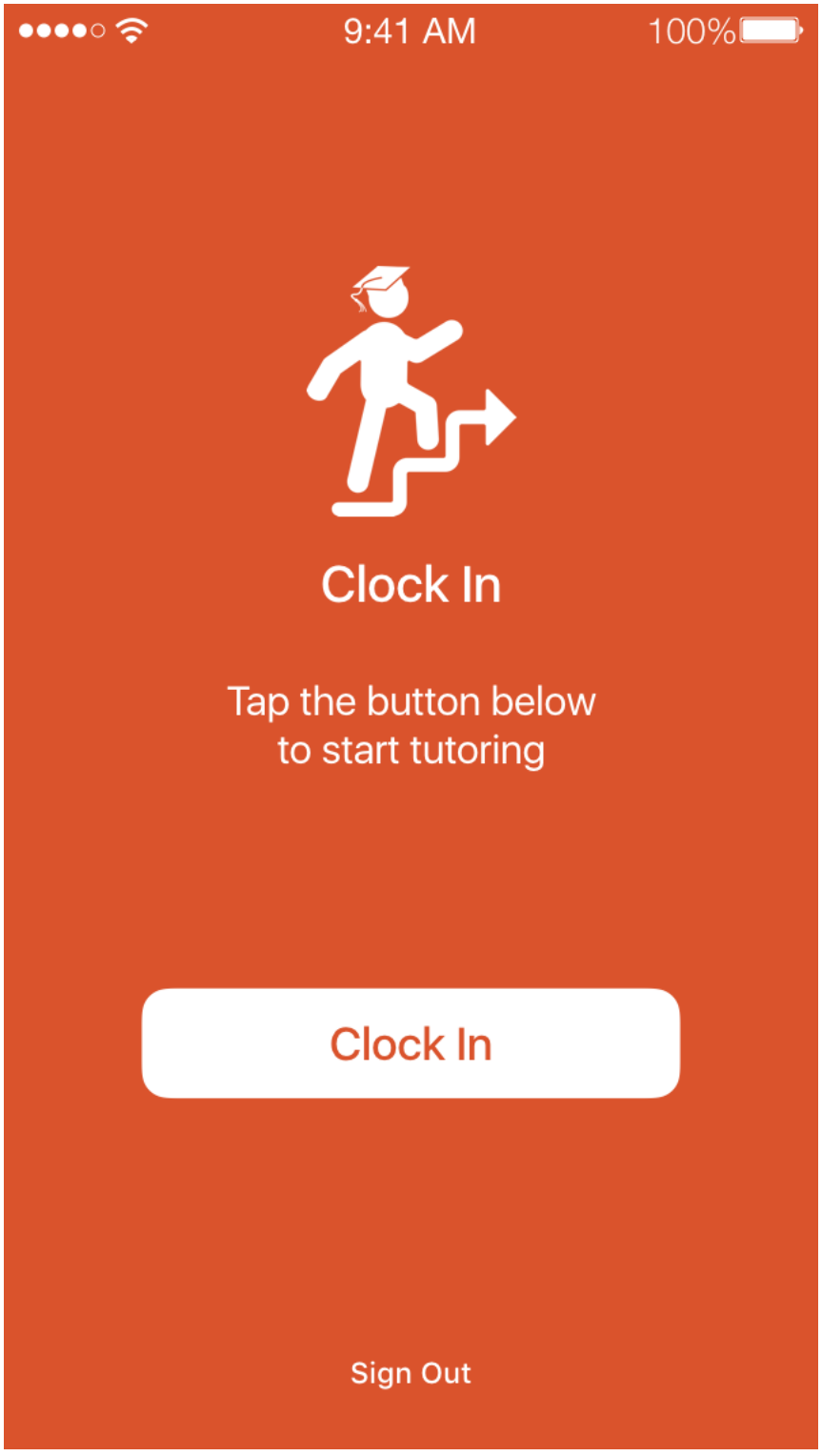


Figure 15: Clock On

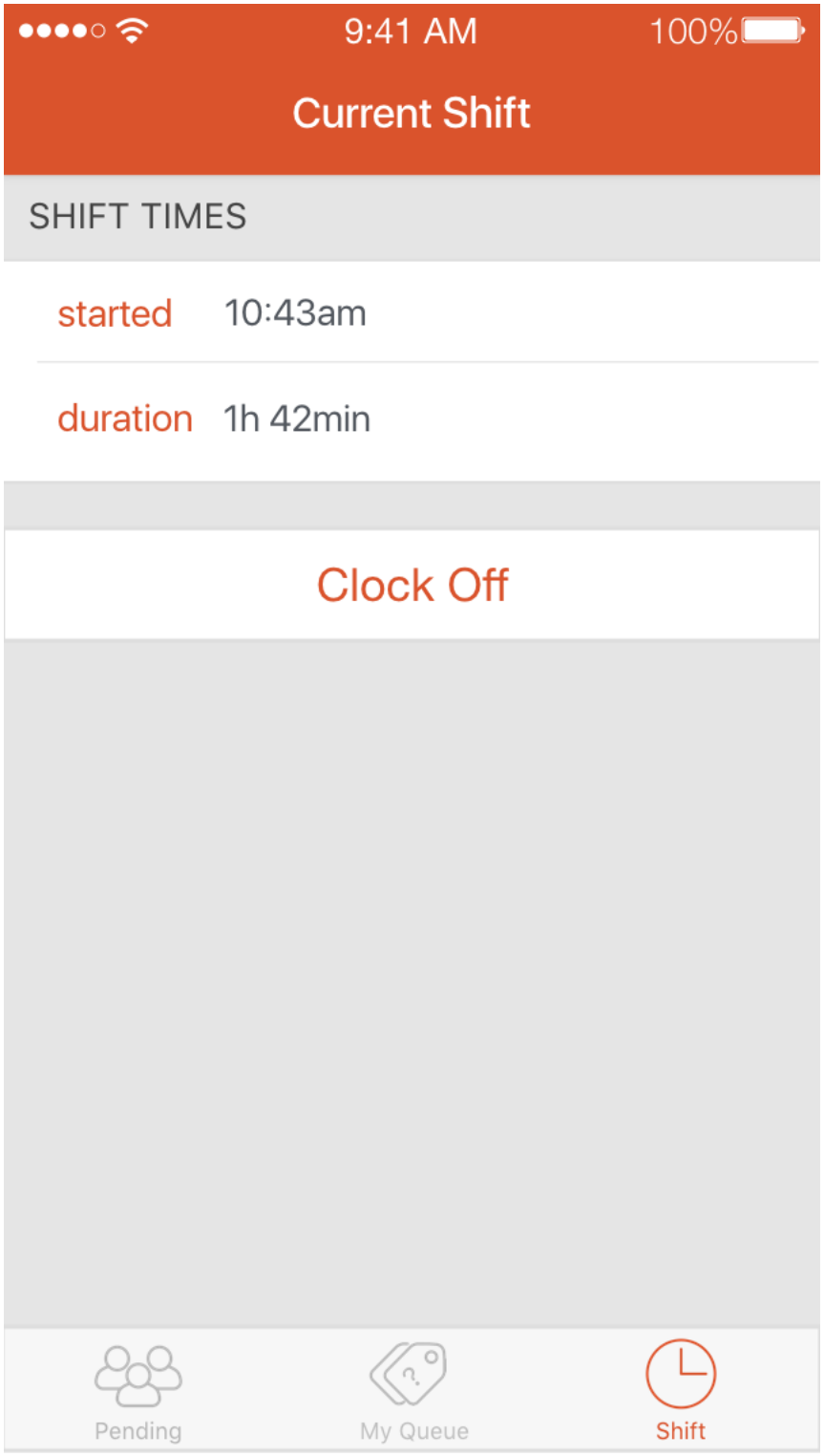


Figure 16: Shift Tab

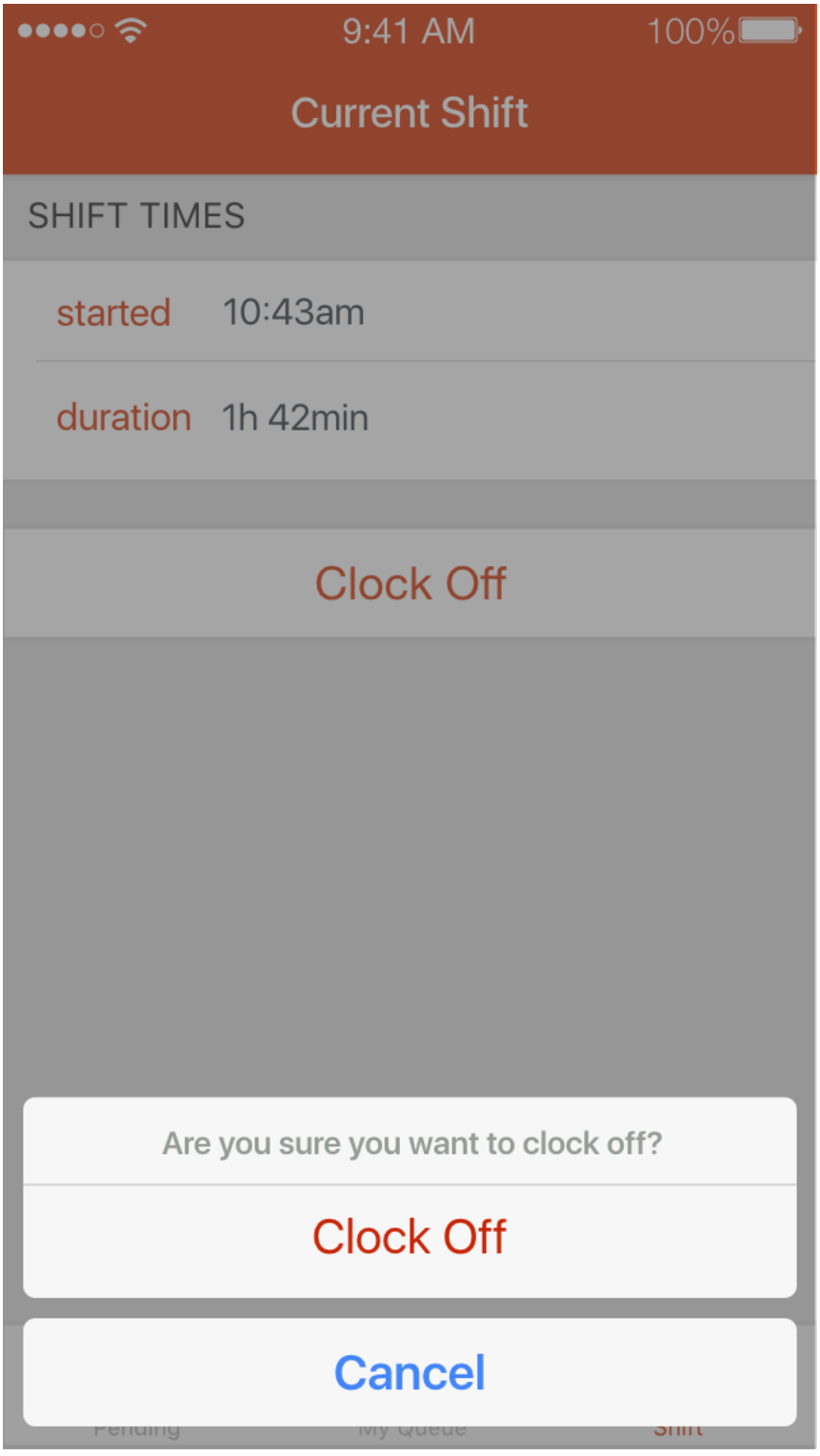
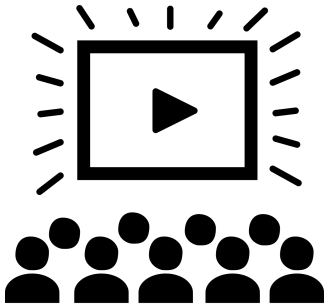


Figure 17: Shift Tab - Clocking Off



Chapter 11

Presentation Video

Storyboard and Links

Contents

1 Celtx

Link to Celtx Storyboard[1]. Requires login.

Storyboard available as downloadable PDF[2].

2 Final Copy

Download final copy from Dropbox[3].

[1] See https://www.celtx.com/a/ux/story?id=https%3A%2F%2Fwww.celtx.com%2Ffeeds%2Fdefault%2Fprivate%2Ffull%2F3cb946b19eaeb19048e0ca8b76622da7c07cc09d

[2] See https://www.dropbox.com/s/by49djjg40n41hk/Storyboard.pdf?dl=1

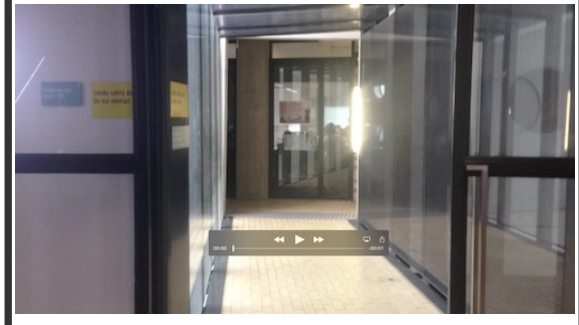
[3] See https://www.dropbox.com/s/znd31uhqno1wd56/FINALFINAL.mp4?dl=0

1. Introduction Context

1.1 WIDE

Time-lapse of ATC stairs - 3 seconds of time-lapse

1.2 MEDIUM



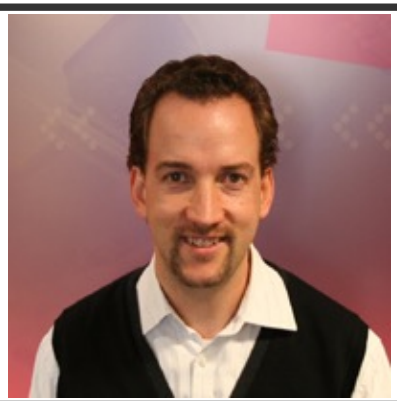
Hyper lapse walking from lobby of Level 6 into ATC620

1.3 MEDIUM

Hold on PHD for a few seconds. Introduce voiceover from Andrew

2. Context Interviews

2.1 MEDIUM CLOSE UP



Andrew describes what the PHD is in a few sentences. Tag his name at the bottom.

2.2 MEDIUM CLOSE UP

Show another tutor describing the helpdesk in a one sentence. x2-3

2.3 CUTAWAY

Show students getting help from tutors

2.4 MEDIUM CLOSE UP

Ask a student:
Describe PHD is useful for you? x2-3

2.5 CUTAWAY

Time-lapse of the helpdesk

3. Problem

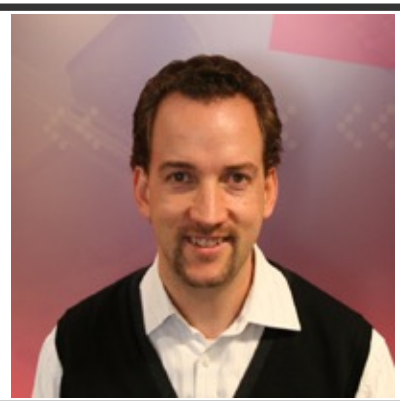
3.1 MEDIUM

Does the helpdesk get busy as a tutor? 1-2 sentences from 1-2 tutors

3.2 MEDIUM

Student's find it hard to get help when it's busy. 1-2 sentences

3.3 MEDIUM



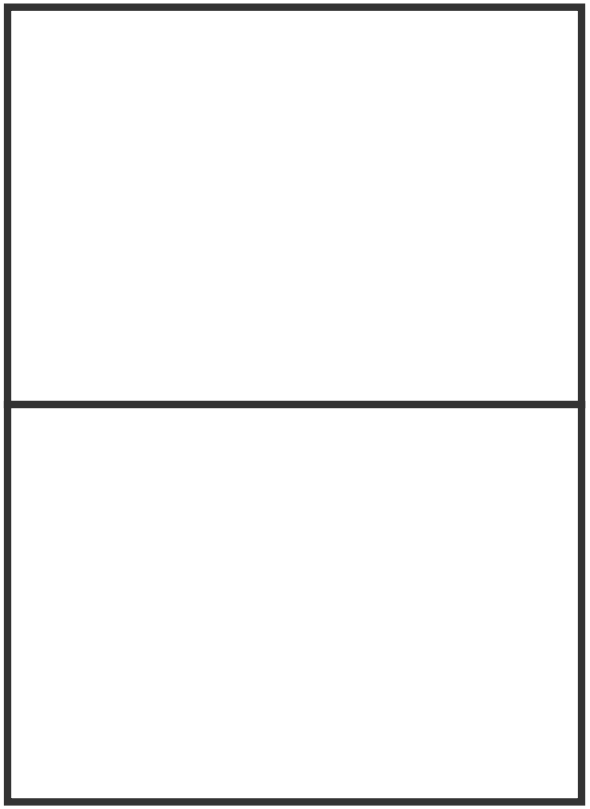
Andrew: are your tutors always there on time? Getting stats on which tasks students need to get help from?

3.4 MEDIUM

Tutors find it hard to keep track of students when it's busy

4. Solution: Ticketing

4.1 POV



Introduce ticketing system. Write dot points on paper fast forwarded.

1. Ticketing system (ticket icon?)
2. Easy to use (happy)
3. Noninvasive (hurt)
4. Don't deter students (happy?)

Pause between shots. Allow for voiceover.

5. Benefits

5.1 MEDIUM

Tracking who comes to the helpdesk when. Portfolio versus assignment-based.

5.2 MEDIUM

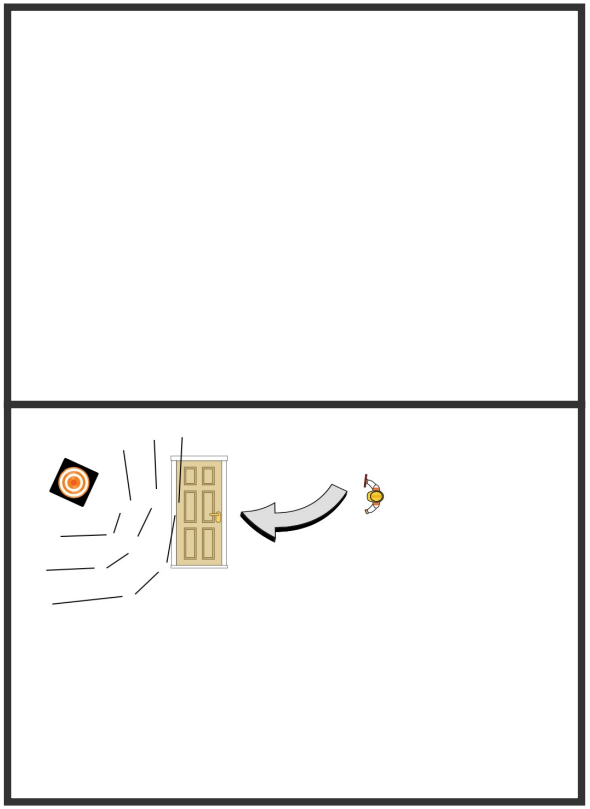
Data: which tutors are on time, and which are not.

5.3 WIDE

Data: students who need help with specific work

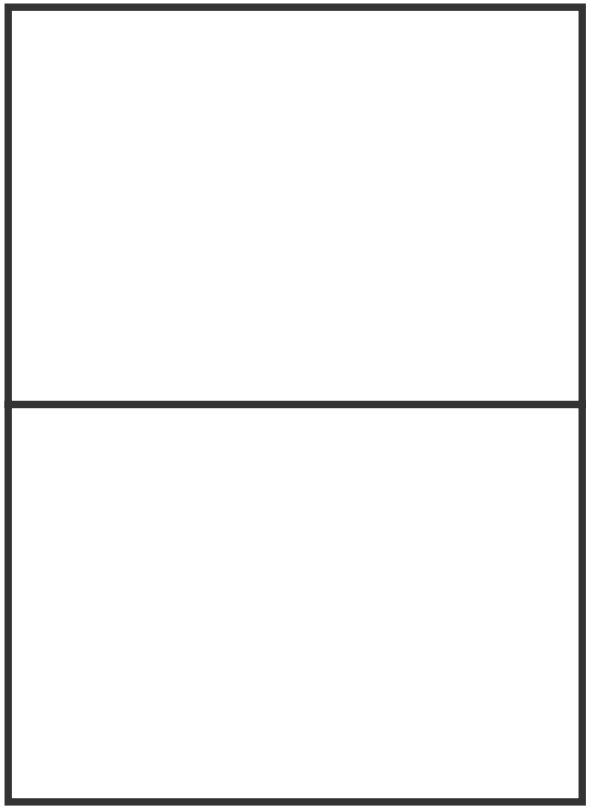
6. Implementation

6.1 POV



Deployment into Helpdesk via beacons

6.2 POV



Demo the iPhone and Android app

Initially start with an empty topdown shot and place each phone down on the desk.

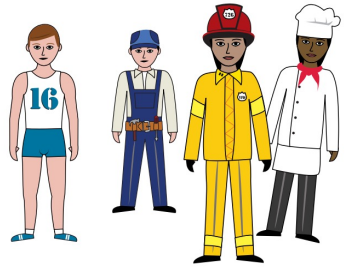
Each phone starts with the Android and iOS lock screen.

Get a push notification.

Run through basic of the prototype.

7. Conclusion

7.1 WIDE



Shot in the helpdesk of us four.

We're the doubt fire helpdesk ticket system team. Our prototype is done and has undergone usability testing. We're to keen to get started

7.2 MEDIUM

Doubtfire Helpdesk

Show doubt fire helpdesk logo

Chapter 12

Alex's Worklog

Contribution Statements

Contents

1 Week 1

Date: February 29 to March 4

1.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	3
Research	0
Software	10
Presentations	0
Team Documents	2
Reviewing	0
Other	0
Total Hours This Week	16

1.2 Detail of tasks undertaken

1.2.1 Initial Group Meeting

29 Feb 12:30pm

- Attended first final year project lecture
- Finalised team group members
- Met other Doubtfire team

1.2.2 Set Up Trello Boards

29 Feb 5:30pm

- Created Doubtfire Trello Board for Helpdesk Ticketing System[1]
- Invited relevant team members to the board
- Added all tasks for final year project into Upcoming column

[1] See `http://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`

1.2.3 Created the Final Year Project GitHub and Wiki

29 Feb 3:30pm

- Created the final year project wiki on GitHub[2]
- Created team contact page[3]
- Created minutes template[4]
- Put the project deliverables[5] up on this Wiki
- Forked the Doubtfire repositories onto the `final-year-project` team

1.2.4 Set up Slack Channels

29 Feb 5:30pm

- Invited all team members to the existing Doubtfire Slack team[6]
- Created relevant channels for the team:
- `#final-year-projects` - Any general discussion for students working on Doubtfire final year projects
- `#helpdesk-ticketing` - Discussion for the help-desk system final year project

1.2.5 Refactor Doubtfire Web Front End

Ongoing throughout the week - at least 10 hours

- Necessary code refactor for the Doubtfire front-end
- Andrew Cain wanted this to be done so that the new students working on the project have an improved codebase to start working on (modularise the codebase into smaller components)
- Pushed 129 commits. See Pull Request #8[7]

1.3 Learning

N/A

[2]See `http://github.com/final-year-project`
[3]See `http://github.com/final-year-project/wiki/Group-Contact-Details`
[4]See `http://github.com/final-year-project/wiki/Minutes-Template`
[5]See `https://github.com/final-year-project/documentation/wiki/Deliverables`
[6]See `http://doubtfire.slack.com`
[7]See `http://github.com/doubtfire-lms/doubtfire-web/pull/8`

1.4 Collaboration

- Worked with team, lots of communication on the Slack team
- Decided what needed to be done on the immediate timeline
- A lot of meta work and team-based administration this week; has involved a lot of team communication

---

2 Week 2

Date: March 7 to March 11

2.1 Summary

Task	Hours
Team and Supervisor Meetings	2
Client Meetings	1
Team Management and Administration	1
Research	0
Software	3
Presentations	0
Team Documents	2
Reviewing	0
Other	0
Total Hours This Week	10

2.2 Detail of tasks undertaken

2.2.1 Initial Meeting with Graham Farrell

7 Mar 1:00pm

- Met up for about 15 minutes with Graham

- See meeting minutes[8] for more details

2.2.2 Initial Meeting with Andrew Cain

7 Mar 1:15pm

- Met up for about 1 hour with Andrew and team
- See meeting minutes[9] for more details

2.2.3 Spent time trying to add Docker for Windows developers

7 Mar - Various times throughout the day

- As developing Doubtfire must be done on a UNIX system, I tried to look into integrating Docker[10] into Doubtfire
- Created two Docker branches on the API and Web repositories (`config/add-docker`)
- See commit changes:
- Web - 4 commits[11]
- API - 13 commits[12]

2.2.4 Update Doubtfire contributing documents

7 Mar 8:30pm

- As an action item from the client meeting this week (see Follow Up Item 5[13]) I was to update the contributing documents and improve the Git workflow process documentation and coding convention
- See commit changes:

[8] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-7-mar-at-100pm`

[9] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-7-mar-at-115pm`

[10] See `http://docker.com`

[11] See `https://github.com/doubtfire-lms/doubtfire-web/compare/13873ca4439532f98c7d11f984480103ddb0ff88...ac6bfa2b5fe70b129ea55332f0d01dfa101daeea`

[12] See `https://github.com/doubtfire-lms/doubtfire-api/compare/662cc4cd369cf37ff9ccb5e16e11e7bc8d748e16...35e6447742d1f380787c167033e3ca8f0baefcb5`

[13] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#follow-up-actions-1`

- Web - 2 commits[14]
- API - 2 commits[15]

2.2.5 Meeting to discuss Trello workflow

9 Mar 4:30pm

Refer to Meeting Minutes[16] for more on this item.

2.2.6 Write up Trello Workflow and SDLC Plan

13 Mar 3:30pm

Write up SDLC plan[17] notes

2.3 Learning

N/A

2.4 Collaboration

- Team meetings both client and supervisor
- Chat on Slack
- Updates to Trello board

---

3 Week 3

Date: March 11 to March 18

3.1 Summary

[14]See https://github.com/doubtfire-lms/doubtfire-web/compare/ecaf5041948af2349b8c06f74e900435a4e22809...cc599df9e34b8fd60c0fdfa51b74d235f1c1f7e1

[15]See https://github.com/doubtfire-lms/doubtfire-api/compare/8768dc72c289d2ec55f195d74312f34067674815...46906fd6d5e9b1004cec821b8c2f6fdaf0f107f5

[16]See https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-9-mar-at-430pm

[17]See https://github.com/final-year-project/documentation/wiki/SDLC-Plan

Task	Hours
Team and Supervisor Meetings	2
Client Meetings	0
Team Management and Administration	2
Research	0
Software	6
Presentations	0
Team Documents	0
Reviewing	2
Other	0
Total Hours This Week	12

3.2 Detail of tasks undertaken

3.2.1 Development session with team

12 Mar 1:30pm

- Met with Jake and Reuben to work on some tasks together
- Reviewed a lot of work and gave feedback to them on their Pull Requests

3.2.2 Assisting Declan and Lachlan with Doubtfire installs

12 Mar 1:30pm, 15 Mar 1:30pm

- Assisted both Lachlan and Declan to install Doubtfire over slack
- Helped Declan on the 15th to get Doubtfire running but we didn't succeed

3.2.3 Stylesheet Refactor

15 Mar 8:30pm

- Submitted Pull Request #14 to Doubtfire Web[18] that refactors all the stylesheets to use SASS over LESS

[18] See `https://github.com/doubtfire-lms/doubtfire-web/pull/14`

3.2.4 Sprint 1 Retrospective

Mar 20 1:30pm

- Ran a retrospective over Skype for Sprint 1
- Refer to meeting minutes[19]

3.3 Learning

- Brush up on SASS[20] skills

3.4 Collaboration

- Team meetings
- Chat on Slack
- Chat over Skype
- Updates to Trello board

---

4 Week 4a

Date: March 21 to March 26

4.1 Summary

Task	Hours
Team and Supervisor Meetings	3
Client Meetings	1
Team Management and Administration	1
Research	0
Software	6

[19] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-20-mar-at-130pm`

[20] See `http://sass-lang.com`

Task	Hours
Presentations	0
Team Documents	0
Reviewing	3
Other	0
Total Hours This Week	14

4.2 Detail of tasks undertaken

4.2.1 Meeting with Graham

Mar 21 1:15pm

- Refer to meeting minutes[21]

4.2.2 Meeting with Andrew Cain

21 Mar 1:30pm

- Refer to meeting minutes[22]

4.2.3 Assisting Jake and Reuben with Minitest

21 Mar 4:30pm

- Spent a few hours helping Jake and Reuben on setting up Minitest for the API
- Managed to get a few tests up and running
- Assisted with sharing prior knowledge of Ruby and Rails tests

4.2.4 Minor improvements to UI and refactor build pipeline

Mar 22 10:30am

- Wrote 19 commits on improving the build pipeline

[21] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-20-mar-at-115pm`

[22] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-20-mar-at-130pm-1`

- Drastically improved the speed at which changes are reloaded whilst developing
- Refer to these 19 commits pushed to GitHub[23]

4.2.5 Improve documentation

24 Mar 8:30pm

- Submitted changes to documentation that improve workflow
- Some group members found the workflow documentation too verbose for quick reference, so summary sections were added
- See these three commits[24]

4.3 Learning

- Read about Grunt.js[25] build pipeline

4.4 Collaboration

- Team meetings
- Chat on Slack
- Updates to Trello board

---

5 Week 4b

Date: March 27 to April 1

5.1 Summary

[23] See `https://github.com/doubtfire-lms/doubtfire-web/compare/59e16ecb18a7e7ca76e2b7d726e61b4088624d28...029d539e4ab5430233f68c287317dab7211b74ab`

[24] See `https://github.com/doubtfire-lms/doubtfire-api/compare/96aa913e20b1116440ddfe39d9afe4cfe47536ec...04957d5b5d025f69b81ff172d6bc36dd0f316c8c`

[25] See `http://gruntjs.com`

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	1
Research	0
Software	5
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	7

NB: This week was the Easter holiday break

5.2 Detail of tasks undertaken

5.2.1 Skype meeting

Mar 27 1:30pm

- Refer to meeting minutes[26]

5.2.2 Updates to API Wiki

28 Mar 1:30pm

- Wrote documentation on contributing[27] via Trello
- Wrote developer FAQs[28] on wiki

[26] See https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-28-mar-at-130pm

[27] See https://github.com/doubtfire-lms/doubtfire-api/wiki/Trello-Workflow/04732891b805e318b12d85bfe8b19c0bd59f2088

[28] See https://github.com/doubtfire-lms/doubtfire-api/wiki/Development-FAQs/10f5ffe59c126b9b849e06cb5310da92f502b420

5.2.3 Start working on refactoring Portfolio Wizard

30 March

- Rework the Portfolio Wizard to be more user-friendly
- Remove out the old file upload code and replace with new `file-uploader` directive
- See these three commits[29]

5.3 Learning

- N/A

5.4 Collaboration

- Skype meetings
- Updates to Trello board

---

6 Week 5

Date: April 1 to April 7

6.1 Summary

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	1
Research	0
Software	3
Presentations	0
Team Documents	3
Reviewing	5

[29] See `https://github.com/final-year-project/doubtfire-web/compare/5fd6250...07b8318`

Task	Hours
Other	0
Total Hours This Week	12

6.2 Detail of tasks undertaken

6.2.1 Working with Jake and Reuben on Unit Testing

- Spent time on Monday trying to set up Minitest with Jake and Reuben
- We worked on a couple of hours, refactoring the `seeds.rb` file and adding in the new folder structure required by Minitest
- Added in some initial tests to ensure that Minitest was properly "plugged in" with Rails

6.2.2 Expanding Comment Box Enhancement

4 April

- Worked with Jake to help him with CSS issues on expanding comment box enhance
- See the pull request[30] that we submitted

6.2.3 Improve documentation

5 April

- Update documents for running PDF generation on Doubtfire
- Some group members were having issues with PDF generation due to poor install documentation
- See these two commits[31]

6.3 Learning

- N/A

[30] See `https://github.com/doubtfire-lms/doubtfire-web/pull/27`

[31] See `https://github.com/doubtfire-lms/doubtfire-api/compare/96aa913e20b1116440ddfe39d9afe4cfe47536ec...04957d5b5d025f69b81ff172d6bc36dd0f316c8c`

6.4 Collaboration

- GitHub Pull Requests
- Conversations with team on Monday's lecture times
- Slack
- Updates to Trello board

---

7 Week 6

Date: April 8 to April 14

7.1 Summary

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	1
Research	0
Software	5
Presentations	0
Team Documents	0
Reviewing	5
Other	0
Total Hours This Week	11

7.2 Detail of tasks undertaken

7.2.1 Fix logger class

10 April

- Fixed the Doubtfire logger class to print `fatal` and `error` messages to the console

- See the pull request[32] that was submitted

7.2.2 Improve timeouts occurring on Doubtfire

10 April

- Timeouts occurring on large tasks were taking up way too much time
- Sometimes these timeouts would result in bad errors produced to the user
- See the pull request[33] that was submitted to resolve this

7.2.3 Worked with Jake and Reuben for testing

11 April

- Jake and Reuben were having trouble with writing tests
- We spent a few hours trying to make new helpers for the tests, namely `AuthHelper` and `JsonHelper`
- Needed to spend some time with Andrew to work this one out

7.3 Learning

- N/A

7.4 Collaboration

- Sitting together with Jake and Reuben and working on problem
- Slack
- Updates to Trello board

---

8 Week 7

Date: April 15 to April 21

[32] See `https://github.com/doubtfire-lms/doubtfire-api/pull/11`
[33] See `https://github.com/doubtfire-lms/doubtfire-api/pull/12`

8.1 Summary

Task	Hours
Team and Supervisor Meetings	0.25
Client Meetings	0.75
Team Management and Administration	0
Research	11
Software	0
Presentations	0
Team Documents	2
Reviewing	0
Other	0
Total Hours This Week	14

8.2 Detail of tasks undertaken

8.2.1 Prototyping session with Jake

15 April - 10:30am to 7:30pm

- Worked on prototyping iOS and Android helpdesk companion applications
- Developed basic use cases[34] and general walk-through through the app
- Second iteration of the architecture diagram[35]

8.2.2 Meeting with Graham

18 April - 1:15pm

- Discussed peer reviews
- Graham approved assessment criteria[36]

[34] See https://github.com/final-year-project/documentation/wiki/Requirements-Documentation#use-cases

[35] See https://github.com/final-year-project/documentation/wiki/Requirements-Documentation#high-level-architecture-diagram

[36] See https://github.com/final-year-project/documentation/wiki/Requirements-Documentation#high-level-architecture-diagram

8.2.3 Meeting with Andrew

18 April - 1:30pm

- Feedback from first prototype
- Discussed improvements to improve workflow of the app
- Refer to meeting minutes

8.2.4 Implement second iteration of iOS prototype

18 April - 5:30pm to 8:30pm

- Implement feedback based on meeting with Andrew on 18 April
- Add in new tabs and other extra features into design

8.3 Learning

- Indesign UI prototyping tool

8.4 Collaboration

- Whiteboarding with Jake
- Using Indesign to complete prototypes
- Slack
- Updates to Trello board

---

9 Week 8

Date: April 22 to April 28

9.1 Summary

Task	Hours
Team and Supervisor Meetings	0

Task	Hours
Client Meetings	0
Team Management and Administration	1
Research	0
Software	0
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	1

9.2 Detail of tasks undertaken

9.2.1 General updates to Trello boards and status updates from everyone

**Throughout the week*

- Regularly get status updates and organise Trello board*

9.3 Learning

- N/A

9.4 Collaboration

- Slack
- Updates to Trello board

---

10 Week 9

Date: April 29 to May 5

10.1 Summary

Task	Hours
Team and Supervisor Meetings	3
Client Meetings	0
Team Management and Administration	0
Research	0
Software	6
Presentations	3
Team Documents	1
Reviewing	0
Other	0
Total Hours This Week	13

10.2 Detail of tasks undertaken

10.2.1 Storyboarding session for project video

29 April - 9:30am to 12:30pm

- Worked on storyboard for final year project video
- Refer to Presentation Video[37]

10.2.2 Update Portfolio Wizard

29 April - 9:30am to 2:30pm

- Refer to commits[38]

10.2.3 Film Stock Footage

2 May - 1:30pm to 3:30pm

- Film stock footage for video with Jake

[37] See `https://github.com/final-year-project/documentation/wiki/Presentation-Video`
[38] See `https://github.com/doubtfire-lms/doubtfire-web/compare/29b0fbf...0432a1b`

10.2.4 Update Requirements Documentation

4 May - 1:30pm to 2:30pm

- Update requirements documentation
- New use case descriptions with new workflow

4 May - 2:30pm to 3:30pm

- Update Invision prototype with new screenshots

10.3 Learning

N/A

10.4 Collaboration

- Using Indesign to complete prototypes
- Slack
- Updates to Trello board

---

11 Week 10

Date: May 6 to May 12

11.1 Summary

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	0
Research	0
Software	0
Presentations	0

Task	Hours
Team Documents	2
Reviewing	2
Other	3
Total Hours This Week	7

11.2 Detail of tasks undertaken

11.2.1 Project Planning Document

7 May - 10:30am to 12:30pm

- Begin working on sections of project plan

11.2.2 Begin Compiling Portfolio

10 May - 12:30am to 2:30pm

- Initiate compiling resources together for portfolio

11.3 Learning

N/A

11.4 Collaboration

- Slack
- Updates to Trello board

---

12 Week 11

Date: May 13 to May 19

12.1 Summary

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	1
Research	0
Software	0
Presentations	9
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	10

12.2 Detail of tasks undertaken

12.2.1 Filming With Jake

14 May - 1:00pm to 7:30pm

- Worked with Jake to finish filming of video
- Review editing and give feedback
- Record voiceovers

12.2.2 Presentations

23 May - 12:30pm to 2:30pm

- Presentations for Final Year Project

12.3 Learning

- Adobe Premiere

12.4 Collaboration

- Slack
- Updates to Trello board

- Meetings with Jake

---

13 Week 12

Date: May 20 to May 27

13.1 Summary

Task	Hours
Team and Supervisor Meetings	0.5
Client Meetings	0
Team Management and Administration	0
Research	0
Software	0
Presentations	0
Team Documents	7.5
Reviewing	0
Other	0
Total Hours This Week	8

13.2 Detail of tasks undertaken

13.2.1 Meeting with Graham

25 May - 1:15pm - 1:45pm

- Discussed submission protocol
- Refactor parts of requirement documentation

13.2.2 Convert wiki to LaTeX document

25 May - 7:30pm - 10:30pm, 26 May - 8:00pm - 11:30pm

- LaTeXify Wiki for submission

13.2.3 Work on SRS and Self Reflection

25 May - 7:30pm - 10:30pm, 26 May - 8:00pm - 11:30pm

- Started filling out templates provided on BB
- Used google docs to work out collaboratively
- Convert to wiki ASAP

13.3 Learning

N/A

13.4 Collaboration

- Slack
- Google Docs

Chapter 13

Jake's Worklog

Contribution Statements

Contents

1 12 Week Summary

1.1 Overview

Week	1	2	3	4	5	6	7	8	9	10	11	12
Team and Supervisor meetings	1	2	2	3	1	0	0	1	1	3	0	0
Client Meetings	0	1	0	1	0	0	0	1	0	0	0	2
Team Management and Admin	2	1	1	0	0	0	0	1	0	0	0	0
Research	3	1	4	2	7	3	4	11	0	0	0	0
Software	3	3	12	5	5	5	7	0	0	6	0	1
Presentations	0	0	0	0	0	0	0	0	0	0	0	0
Team Documents	0	1	0	0	2	0	0	2	0	1	1	4
Reviewing	0	0	0	0	4	2	1	0	0	0	0	1
Other	4	0	0	0	0	0	0	0	5	4	20	0
Total	10	10	19	11	18	10	12	16	6	13	21	8

1.1.1 Week 1

[29 Feb 10:30am]

- Setup and finalise development environment
- Setup finalise git envioronment
- Relatively small contribution to doubtfire-web refactoring on branch api-refactor.
- Meetings with Alex to organise work on refactor and development environment.

1.1.2 [Lecture]

[29 Feb 12:30pm]

- Attended Final Year Project lecture
- Met Doubtfire team

1.1.3 [Development]

[02 March 5:00pm]

- Assign myself Calendar-Width trello card and implement fix

1.2 Learning

I familiarised myself with the Doubtfire development environment and utilised some new frameworks for the osx terminal to increase productivity in development. Started familarising myself with the codebase by addressing the calendar width UI card.

1.3 Collaboration

Worked heavily with Alex C to prepare the development environment, run through ideas for the project, met the rest of our Doubtfire team, and the other Doubtfire team.

1.3.1 Week 2

- Worked on researching and learning Doubtfire System
- Work on learning Doubtfire development process
- Finish Calendar card

1.4 Learning

- Continued familiarising myself with the Doubtfire development environment, learned the workflow process for uploading code

1.5 Collaboration

- Again worked with Alex heavily as he already knows the system.

1.5.1 Week 3

- Met with Alex and Reuben to work on some tasks together as well as work on submission process and merge request guidelines
- Skype meeting to discuss sprint.

1.5.2 Week 4

- Meeting Minutes (https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-20-mar-at-115pm)
- Spent a few hours setting up Minitest for the API
- Managed to get a few tests up and running

1.5.3 Week 5

- After deciding on Minitest framework, Reuben, Alex and I all worked on making sure our test suit was setup properly.
- Worked on comment submission with Alex, but a lot of solo learning with this task. (https://github.com/doubtfire-lms/doubtfire-web/pull/27)

1.5.4 Week 6

- Worked with Alex and Reuben writing tests
- We spent a few hours trying to make new helpers for the tests, namely AuthHelper and JsonHelper
- Worked with Andrew as well.

1.5.5 Week 7

1.5.5.1 Collaboration

- Working with Alex to create the IOS prototypes

1.5.5.2 Learning/progress

- Huge progress with creating the prototypes for Android
- Used Invision to setup high fidelity prototype presentaiton

---

1.5.6 Week 8

- Work and help Alex manage the Slack and Trello

1.5.7 Week 9

- Created and planned storyboard for presentation video
- Film stock footage
- update Prototypes after talking with Andrew on use case changes

---

1.5.8 Week 10

- Worked on adding email button to plagiarism view for tutors
- Worked with Andrew on method to ensure that only tutors see button.

1.5.9 Week 11

- Presented video in lecture
- Worked with Alex to finish filming of video
- Finished all the editing
- Record voiceovers

1.5.10 Week 12

- Finished SEP document with Reuben over skype
- use slack to finalize documentation
- Worked on self-reflection documents.
- Slack management

Chapter 14

Lachlan's Worklog

Contribution Statements

Contents

1 12 Week Summary

1.1 Overview

Week	1	2	3	4	5	6	7	8	9	10	11	12
Team and Supervisor meetings	1	2	1.5	1.8	1	1	1.8	4	4	3	1	1.5
Client Meetings	0	1	0	1	0	0	0.2	0	0	0	0	0
Team Management and Admin	1	0	0	0	0	0	0	0	0	0	0	0
Research	1	5	2	0	1	2	0.7	0	0	0	0	0
Software	0	0	3	10	4	2.5	3	1	0	0	0	0
Presentations	0	0	0	0	0	0	0	0	0	0	1	0
Team Documents	0	0	0	0	0	2.5	0	0	0	0	0	4
Reviewing	0	0	0	0	0	0	1	1	0	0	0	0
Other	0	0	0	0	0	0	0	0	0	0	0	0
Total	3	8	6.5	12.8	6	8	6.7	6	4	3	2	5.5

2 Week 1

Date: February 29 to March 6

2.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	1
Research	1
Software	0
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	3

Task	Hours

2.2 Detail of tasks undertaken

29/02

- Software Project lecture.
- Met with team to discuss the parameters of the project and technologies involved.

29/02 - 4/03

- Researched VM technolgies that I might need
- Attempted to download VM image provided by Alex (there were issues)

2.3 Learning

- I got a high level view of how doubtfire is structured behind the scenes as well as the technologies involved.
- Attending the lecture I learned the outline of the unit and the basics of what I'll need to do throughout the semester.

2.4 Collaboration

- I participated in a team meeting to discuss what we're going to do for this project.
- Alex and I discussed what might work in terms of using a VM instead of installing Ubuntu, for setting up the dev environment.

---

3 Week 2

Date: March 7 to March 13

3.1 Summary

Task	Hours
Team and Supervisor Meetings	2
Client Meetings	1
Team Management and Administration	0
Research	5
Software	0
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	8

3.2 Detail of tasks undertaken

7/03

- Software Project lecture.
- Met with our client, Andrew Cain.

8/03

- Researched AngularJS, one of the main technologies being used for doubtfire (web).

7/03 - 13/03

- Switched from attempting VM usage to dual-boot Ubuntu on my laptop.
- Researched how to safely set up a dual boot and installation options for Ubuntu.
- Met with the team to discuss GitHub workflow and the wider context of how we should do our work.
- Set up laptop with dual-boot Ubuntu.
- With mixed success, installed required software to set up my dev environment in Ubuntu. Collaborated Alex who helped me solve numerous errors I encountered during this process.

3.3 Learning

- During the lecture, I learned about how to be professional when dealing with a client.
- I learned a great deal about GitHub workflow, including forking and pull requests.
- I learned more detail about what Andrew would like for the doubtfire system, and how it fits in to existing features.

3.4 Collaboration

- I collaborated with my team during the client meeting and also in the team meeting to discuss GitHub workflow.
- Worked with Alex to solve my many installation errors setting up my environment.

---

4 Week 3

Date: March 14 to March 20

4.1 Summary

Task	Hours
Team and Supervisor Meetings	1.5
Client Meetings	0
Team Management and Administration	0
Research	2
Software	3
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	6.5

4.2 Detail of tasks undertaken

14/03 - 20/03

- Got my laptop development environment up and running and started looking through the code.
- Began working on a trello card job to make some changes to the front end Card[1]

4.3 Learning

- During the lecture, I learned about how requirements analysis is viewed in context of the capstone projects.
- Did some more of the AngularJS tutorial to get a better handle of the language features.

4.4 Collaboration

- Had a team meeting on the Sunday of week 3 to discuss the requirements for the project, the scoping for each semester and what issues we need to bring up at the week 4 meetings with our supervisor (Graham) and our client (Andrew).

5 Week 4

Date: March 21 to April 3

5.1 Summary

Task	Hours
Team and Supervisor Meetings	1.75
Client Meetings	1
Team Management and Administration	0
Research	0
Software	10

[1] See `https://trello.com/c/QSAWPy36/35-add-in-alt-keys-using-accesskey-attribute-to-quickly-perform`

Task	Hours
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	12.75

5.2 Detail of tasks undertaken

28/03

- Team meeting on skype to discuss requirements

21/03 - 03/04

- Worked on Accessor Keys[2] task. Ended up passing it back to the backlog as it proved too difficult.
- Worked on Adding Grade Icon to Task Viewer[3]
- Meeting with Andrew to discuss requirements.
- Short meeting with Graham to discuss our progress and our assessment criteria.

5.3 Learning

- I learned more about how services work in AngularJS while doing the Grade Icon task
- I learned more about how the HTML pages integrate with AngularJS

5.4 Collaboration

- Team meeting, client meeting and getting assistance from Alex on the 2 cards I worked on.

[2]See `https://trello.com/c/QSAWPy36/35-add-in-alt-keys-using-accesskey-attribute-to-quickly-perform`
[3]See `https://trello.com/c/5cVfPtyU/39-add-grade-details-to-task-sheet-so-that-students-can-see-wh`

6 Week 5

Date: April 4 to April 10

6.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	0
Research	1
Software	4
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	6

6.2 Detail of tasks undertaken

04/04 - 10/04

- Began working on a task to fix a Latex issue when generating pdfs. card[4]

6.3 Learning

- Spent some time learning about the basics of Latex

6.4 Collaboration

- Participated in informal meeting with Andrew on Monday, where we went through some test harness issues.

[4]See `https://trello.com/c/Ru6IuDnN/42-m-text-in-file-upload-requirements-can-break-latex-processir`

7 Week 6

Date: April 11 to April 17

7.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	0
Research	2
Software	2.5
Presentations	0
Team Documents	2.5
Reviewing	0
Other	0
Total Hours This Week	8

7.2 Detail of tasks undertaken

11/04 - 17/04

- Finished API task with help from Andrew (card[5])
- Started work on new card[6], getting a handle on how CSV import process works on the backend.
- Worked on Use Case Diagram, use Case descriptions and an Entity Model for the ticketing system (card[7])
- Did my mid-semester peer review (card[8])

[5]See `https://trello.com/c/Ru6IuDnN/42-m-text-in-file-upload-requirements-can-break-latex-processir`
[6]See `https://trello.com/c/koISTdFm/46-s-import-of-students-should-use-named-rather-than-positional`
[7]See `https://trello.com/c/RX0UstqJ/8-requirements-documentation`
[8]See `https://trello.com/c/g4R3IkWd/9-peer-review-documents-round-1`

7.3 Learning

- I learned more about how the Rails backend works and how to perform actions and manipulate data using the Rails command line.

7.4 Collaboration

- Talked with Jake and Alex about the requirements documentation and changes needed to make them better.

8 Week 7

- Attended the lecture.
- Further work on import columns card[9]. Put into open pull request.
- Did some research on how Ruby handles various types of arrays.
- Reviewed design documentation created by Alex and Jake, looking for anything that could use improvement.
- Client meeting with Andrew to discuss workflow of the product
- Supervisor meeting with Graham to discuss peers reviews and assessment criteria

9 Week 8

- Attended the lecture.
- Import columns card (see week 7) needed further fixes. Spend some time reviewing the mistakes I made.
- Met with Jake and Alex to story-board the presentation video on a big whiteboard.
- Worked with Alex to attempt to diagnose a bug I found in (live) Doubtfire.

10 Week 9

- Attended the lecture.
- Helped filming stock footage of the help desk.

[9] See `https://trello.com/c/koISTdFm/46-s-import-of-students-should-use-named-rather-than-positional`

11 Week 10

- Attended the lecture.
- Helped Alex and Jake film Andrew, footage we will used in our presentation video.

12 Week 11

- With Alex and Jake, presented our video. Participated in asking questions of the other groups.

13 Week 12

- Lecture
- Met with Graham to discuss Assessment criteria and various documentation requirements.
- Worked on the SEP draft
- Worked on self-reflection documents.

Chapter 15

Reuben's Worklog

Contribution Statements

Contents

1 Week 1

Date: March 29 to April 4

1.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	0
Research	0
Software	0
Presentations	0
Team Documents	0
Reviewing	0
Other	2
Total Hours This Week	3

1.2 Detail of tasks undertaken

1.2.1 Initial Group Meeting

29 Feb 12:30pm

- Attended first final year project lecture
- Finalised team group members
- Met other Doubtfire team

1.2.2 Initial Group Meeting

30 Feb 10:30am

- Set up development environment for DoubtFire Web and API
- Explored DoubtFire project in order to familiarise with product

1.3 Learning

Learned about the tools required in order to work on DoubtFire and basic understanding of how API works

1.4 Collaboration

Worked along side other group members, mainly Alex in order to learn more about the product I'm working on

---

2 Week 2

Date: March 7 to March 11

2.1 Summary

Task	Hours
Team and Supervisor Meetings	2
Client Meetings	1
Team Management and Administration	0
Research	0
Software	1
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	4

2.2 Detail of tasks undertaken

2.2.1 Initial Meeting with Graham Farrell

7 Mar 1:00pm

- Met up for about 15 minutes with Graham
- See meeting minutes[1] for more details

2.2.2 Initial Meeting with Andrew Cain

7 Mar 1:15pm

- Met up for about 1 hour with Andrew and team
- See meeting minutes[2] for more details

2.2.3 Meeting to discuss Trello workflow

9 Mar 4:30pm

Attended Trello session hosted by Alex, refer to Meeting Minutes[3] for more on this item.

2.3 Learning

Learned about Trello and how as a team it will be used as a central reference point in regards to organising sprints and overall project management

2.4 Collaboration

- Team meetings both client and supervisor

---

3 Week 3

Date: March 14 to March 18

[1] See https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-7-mar-at-100pm

[2] See https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-7-mar-at-115pm

[3] See https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-9-mar-at-430pm

3.1 Summary

Task	Hours
Team and Supervisor Meetings	1
Client Meetings	0
Team Management and Administration	0
Research	0
Software	10
Presentations	0
Team Documents	0
Reviewing	0.5
Other	0
Total Hours This Week	11.5

3.2 Detail of tasks undertaken

3.2.1 DoubtFire development work

14 Mar 3:30pm

- Work on quality enhancements for DoubtFire. See Trello card[4] for details

3.2.2 DoubtFire development work

15 Mar 4:00pm

- Work on bug fixes for DoubtFire. See Trello card[5] for details

3.2.3 DoubtFire development work

16 Mar 3:30pm

- Work enhancements and bug fixes DoubtFire. See Trello card[6] for details in regards to enhancements and Trello card[7] for bug fixes

[4]See `https://trello.com/c/oHh2rq9C/29-change-fix-and-include-to-do-not-resubmit-in-ui-and-api-whe`
[5]See `https://trello.com/c/DMk5GeyW/15-students-who-are-already-enrolled-in-a-lab-are-enrolled-twi`
[6]See `https://trello.com/c/YvVVTTPL/34-unit-dropdown-only-shows-unit-name-making-it-impossible-to-`
[7]See `https://trello.com/c/DMk5GeyW/15-students-who-are-already-enrolled-in-a-lab-are-enrolled-twi`

3.2.4 Reviewing - Consultation with project leader

16 Mar 10:30am

- Work with Alex Commaudo to improve erroneous git commits and branching in order to make commits and branching conform to specified development guidelines as outlined in workflow documentation.

3.2.5 Sprint 1 Retrospective

Mar 20 1:30pm

- Ran a retrospective over Skype for Sprint 1
- Refer to meeting minutes[8]

3.3 Learning

By working on minor changes to DoubtFire as outlined above, I learned about Angular and familiarised with Coffeescript. Also, through collaboration with Alex, I learned more about appropriate git workflow.

---

4 Week 4

Date: March 21 to April 01

4.1 Summary

Task	Hours
Team and Supervisor Meetings	1.25
Client Meetings	0
Team Management and Administration	0
Research	0

[8] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#meeting-20-mar-at-130pm`

Task	Hours
Software	15
Presentations	0
Team Documents	0
Reviewing	0.5
Other	0
Total Hours This Week	16.75

4.2 Detail of tasks undertaken

4.2.1 Reviewing - Consultation with project leader

21 Mar 10:30am

- Work with Alex to finalise changes required for pull request to be finalised. Work being merged is specified in Trello card[9]

4.2.2 Meeting with Graham Farrell

21 Mar 1:15pm

- Met with Graham for about 15 minutes

4.2.3 Meeting with Andrew Cain

21 Mar 1:30pm

- Met with Andrew for 1 hour to discuss project details

4.2.4 Development

21 Mar 3:30pm

- Work with Jake and Alex in order to implement Minitest[10] framework. Hit some road blocks in regards to how to best integrate Minitest. Spent most of the time configuring and not writing many tests.

[9] See `https://trello.com/c/DMk5GeyW/15-students-who-are-already-enrolled-in-a-lab-are-enrolled-twic`
[10] See `http://ruby-doc.org/stdlib-2.0.0/libdoc/minitest/rdoc/MiniTest.html`

22 Mar 2:00pm

- Work with Jake and Andrew in order to configure Minitest[11] framework. Got to the point where framework is integrated and working correctly but needs some refinement.

22 Mar 4:00pm

- Work with Andrew on debugging API issue that was causing enrolment to break. Resolved issue. Implemented web features that allow convenor to change student enrolment from convenor view in DoubtFire.

4.2.5 Reviewing - Consultation with project leader

24 Mar 8:30am

- Work with Alex to resolve git work flow issues.

4.2.6 Development

** 25 Mar 2:30pm**

- Work on API and web changes. See Trello card[12].

4.3 Learning

Learned about Minitest and integration of Minitest library into Doubtfire.

---

5 Week 5

Date: April 04 to April 08

5.1 Summary

[11] See `http://ruby-doc.org/stdlib-2.0.0/libdoc/minitest/rdoc/MiniTest.html`
[12] See `https://trello.com/c/uqegS1nt/38-record-and-report-times-submitted-as-well-as-times-assessed`

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	0
Research	3
Software	13
Presentations	0
Team Documents	2
Reviewing	0
Other	0
Total Hours This Week	18

5.2 Detail of tasks undertaken

5.2.1 Software - Consultation with project leader and client

04 Apr 1:30pm

- Work with Alex and Andrew in order to try and resolve ongoing issue with `pdflatex`. Issue is preventing the generation of PDF submissions in development environment via running `rake`

5.2.2 Research - Minitest and API

04 Apr 4:30pmm

- Research how to test Doubtfire API endpoints with minitest. See Trello card[13].

5.2.3 Software - Minitest and API

05 Apr 2:30pm

- Implement API tests for auth end point. See Trello card[14].

[13]See `https://trello.com/c/KYp1ZP0W/43-create-unit-test-for-auth`
[14]See `https://trello.com/c/KYp1ZP0W/43-create-unit-test-for-auth`

5.2.4 Documentation - Draft marking criteria

05 Apr 6:30pm

- Draft marking criteria.

5.2.5 Software - Minitest and API

06 Apr 6:30pm

- Fix integration of Minitest into Doubtfire. See Trello card[15].

5.2.6 Software - Minitest and API

07 Apr 4:30pm

- Start implementing tests for units api endpoint. See Trello card[16]

5.2.7 Software - Minitest and API

10 Apr 12:30pm

- Work with Jake in order to expand on tests for units api endpoint. See Trello card[17]

5.3 Learning

Improve on Minitest knowledge.

---

6 Week 6

Date: April 11 to April 15

6.1 Summary

[15]See `https://trello.com/c/KYp1ZP0W/43-create-unit-test-for-auth`
[16]See `https://trello.com/c/tKRT01L5/45-1-create-unit-tests-for-getting-adding-units`
[17]See `https://trello.com/c/tKRT01L5/45-1-create-unit-tests-for-getting-adding-units`

Task	Hours
Team and Supervisor Meetings	0
Client Meetings	0
Team Management and Administration	0
Research	0
Software	0
Presentations	0
Team Documents	0
Reviewing	0
Other	0
Total Hours This Week	0

6.2 Detail of tasks undertaken

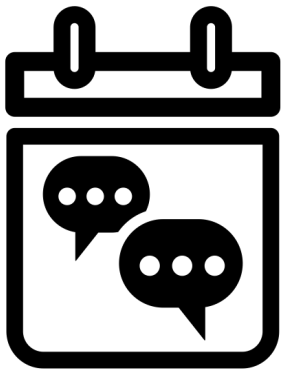
6.2.1 Software - Consultation with project leader and client

11 Apr 1:30pm

- Work with Alex and Andrew in order to try and resolve ongoing issue with `pdflatex`. Issue is preventing the generation of PDF submissions in development environment via running `rake`

6.3 Learning

Improve on Minitest knowledge.



Chapter 16

Meeting Minutes

Contents

1 Meeting 7 Mar at 1:00pm

Location: EN509a

1.1 Present

- Alex Cummaudo
- Jake Renzella
- Reuben Wilson
- Graham Farrell

1.2 Agenda Items

1. Wiki for Documentation
2. Contact Details
3. Meetings with Supervisor

1.3 Discussion, Decisions and Agreements

1.3.1 RE: 1. Wiki for Documentation

- Graham is fine with Wikis as long as he can get physical copies of documents on request

1.3.2 RE: 2. Contact Details

- Graham asked who the team lead is assigned to (Alex)
- Graham wants Alex to send him an email with the list of members in the group *with group photos* ASAP

1.3.3 RE: 3. Meetings

- Graham is happy to meet bi-weekly at around 1:00pm to have a quick catch up on how the project is going
- He is happy for this to change and make it more frequent if need be
- Our meetings can be with Viv if we want also, as long as we email her
- Our next meeting will be in one fortnight (Monday 21 March at 1:00pm)

1.4 Agenda Items For Next Meeting

TBA

1.4.1 Follow Up Actions

1. **Alex** - **ASAP** - Message everyone to put a photo of themselves up on the [[Group Contact Details]] page as well as their details
2. **Jake** - **Friday 19 March** - Confirm meeting with Graham via email for Monday 21 March at 1:00pm
3. **Jake** - **ASAP** - Email Graham our contact emails once (1) is complete

---

2 Meeting 7 Mar at 1:15pm

Location: EN609a

2.1 Present

- Andrew Cain
- Alex Cummaudo
- Jake Renzella
- Reuben Wilson
- Lachlan West
- Declan English
- Doubtfire Interview Booking Team

2.2 Agenda Items

1. Using Windows for Doubtfire Development
2. Interview Booking System
3. Helpdesk Ticketing System
4. Spiking Doubtfire for the next few weeks
5. Discussing Doubtfire Architecture, Testing, and Git Workflow
6. Andrew's Availability and Next Meeting

2.3 Discussion, Decisions and Agreements

2.3.1 RE: 1. Using Windows for Doubtfire Development

- Developing Doubtfire is not supported for Windows
- Developers currently using Windows will require a dual-boot UNIX operating system in order to access Doubtfire code
- Lachlan and Declan will use Ubuntu

2.3.2 RE: 2. Interview Booking System

N/A - not related to our project

2.3.3 RE: 3. Helpdesk Ticketing System

2.3.3.1 Goal

Goal of the system is to track who comes into the helpdesk and what they need help with, on a per-task, per-unit basis

2.3.3.2 User workflow

- Typical end-user (students and tutors) workflow:

1. Students either:
    1. log in to Doubtfire on their computer and create a new helpdesk ticket
    2. use an iPad set up in the helpdesk to indicate that they need help by scanning their student card or entering their student number
2. Students then:
    1. add in what subject they need help
    2. Doubtfire will show them the estimated wait time until a tutor sees them
    3. submit their ticket
3. Doubtfire adds the student to a 'queue' of students waiting to get help
4. While a student is waiting, they may choose to add tags (e.g., `compiling issue`, `bug`, `hand execution`) to their request, and associate the request to a specific Doubtfire task. This will match the right tutor for the right student

5. Tutors receive a push notification of the student on their smartphone granted:
    - they are currently clocked on at the helpdesk
    - they are teaching that subject (unless there are no tutors for the subject currently working)
6. Tutors call out the name of the student who requested help and either:
    1. go and help the student
    2. note that the student did not show up
7. If a student does not show up to their ticket, the tutor will mark that on their app and the student loses reputation. This means:
    - they get a strike and are pushed to the end of the queue
    - on three strikes, they are locked out of the helpdesk ticketing system and cannot make tickets
    - only a tutor or convenor can unlock them

2.3.3.3 Other notes

- Proactive ticket booking (i.e., booking in advance) out of scope at present
- Ticketing system is strictly available to Doubtfire units
- Statistics gathered from help requests will help show difficult or poorly worded tasks (e.g., time taken to help)
- A projector set up in the helpdesk can be used to show:
- the current queue size and people in the queue
- a timeline of the day of how busy the helpdesk has been over the past few hours
- If a student has not submitted extra info (i.e., task they needed help with, concepts/tags) then the tutor will ask them these questions at the end of their ticket before moving to the next student.
- The tutor smartphone app can show students for your units (by default) or the whole queue
- Tutors to clock on to the helpdesk using BLE beacon technology with their smartphone via a Doubtfire app

2.3.4 RE: 4. Spiking Doubtfire for the next few weeks

- As the Product Owner, Andrew Cain wants us to use an Agile/Scrum methodology
- Sprints will be dated fixed
- The first sprint will be from **Mon March 7 to Mon March 21** (3 weeks)
- This sprint will consist of small bug fixes or enhancements that will help the team familiarise themselves with Doubtfire's codebase
- Team will join Trello board and groom a backlog of tasks from the General Development[1] Trello board into the Ticketing System[2] Trello Board
- Alex will merge the backlog of Visualisations tasks into the General Development backlog
- When picking up a task, the team member should size how long they think the task will take to complete (in days)
- The team should see what we think you we get done of these spike tasks in parallel to the other documentation we need to complete for Final Year Project

2.3.5 RE: 5. Discussing Doubtfire Architecture, Testing, and Git Workflow

2.3.5.1 Architecture

Doubtfire's high-level architecture is as thus:

```
Angular JS Web App -> Grape API -> Ruby on Rails Active Record Models -> PostgreSQL D
```

The API is an Ruby on Rails application without Views or Controllers (just Model)

2.3.5.2 Testing

- Unit tests are currently needed ASAP on the Grape API
- UI testing for the Web app is less prioritised
- In the UI, we should opt to move as much code from a service to the UI:
- At present, this code is done in a `modelService` service (e.g., `unitService.addTutorial unit, tutorial`)
- We want to move this to a more object-oritended approach to model functions (e.g., `unit.addTutorial tutorial`)

[1] See `https://trello.com/b/lxTvHiQ9/general-development`
[2] See `https://trello.com/b/8a1k0Wud/helpdesk-ticketing-system`

2.3.5.3 Git Workflow

- Andrew outlined the ideal Git workflow:

1. Fork off `doubfire-lms` (`upstream`) into an organisation (`origin`) for each project (Ticketing, Interview)
2. Branch off `develop` (e.g., `feature/foo`)
3. Make changes
4. Checkout `develop` and pull from `upstream/develop` to get latest changes
5. Merge the branch created in 2 (`feature/foo`) into `develop` and resolve any merge conflicts
6. Push changes to `origin/develop`
7. Add a Pull Request from `origin/develop` to `upstream/develop`
8. Andrew will review the code changes
9. Pull Request will be approved or commented on for improvement

- Alex to merge in the existing pull request from the recent web refactor into `develop`
- Alex to write up an updated `CONTRIBUTING.md` for the Git workflow outlined above and how to set up the upstream and origin (outlined above with images)

2.3.6 RE: 6. Andrew's Availability and Next Meeting

- Andrew will be available from 12:30pm-2:30pm each Monday
- We should aim to meet him at 1:00pm

2.3.7 Additional topics discussed:

- Team to familiarise themselves with administrative side of Doubtfire to use Doubtfire from a non-student perspective

2.4 Agenda Items For Next Meeting

1. Tasks being worked on in relation to familiarity with Doubtfire
2. Requirements analysis based on early progress and new information
3. Questions and answers regarding Doubtfire code

2.4.1 Follow Up Actions

1. **All - By Friday 11 March** - Play around with the Doubtfire from a Convenor perspective
2. **Alex - ASAP** - Merge the Visualisation Trello Board into the General Development Trello Board
3. **Alex - ASAP** - Groom the backlog and pluck out some tasks for us to work on
4. **All - By Monday 21 March** - Get started on some of the tasks from (3) and size up each card (in days) as a comment and begin working on them in a branch
5. **Alex - ASAP** - Merge outstanding PR for structure refactor and update the Doubtfire `CONTRIBUTING.md` to include how to write a Pull Request and the coding convention for the Web App
6. **Declan, Lachlan - ASAP** - Install Ubuntu operating system and set up Doubtfire on it

---

3 Meeting 8 Mar at 2:30pm

Location: N/A - Telephone Call

3.1 Present

- Andrew Cain
- Alex Cummaudo

3.2 Agenda Items

N/A - this was an unplanned telephone call to discussing Doubtfire project management ideas

3.3 Discussion, Decisions and Agreements

- A new board known as the **Doubtfire Backlog Board**[3] which is the Doubtfire *Product Backlog*[4]
- The lists for this board are described as:
- **Fresh Ideas** - New tasks that the product owner thinks of will be added here. But those tasks should be be moved into one of the other lists on this board as soon as possible, and therefore this list should be kept as empty as possible.
- **High Priority** - Tasks that need to be completed ASAP, such as critical bugs
- **Medium Priority** - Tasks that should be completed soon, such as new features or enhancements
- **Low Priority** - Tasks that would be nice to have done eventually, such as small UI beautifications
- **One Day** - Ideas that would be great to have *one day* if we had the time, essentially tasks that are currently too out of scope
- New members joining Doubtfire will move their cards from this board into their own relevant boards for a particular Doubtfire subsystem or project (known as a **Project Board**), or the General Development[5] board if not applicable to a project board.
- On project boards and the General Development board, we have the following columns:
- **Unit Assessment Backlog** - Applicable only to final year project boards and their tasks for the unit's assessment
- **Sprint Backlog** - Tasks involved over the upcoming time-fixed sprint moved from the Doubtfire Backlog board - essentially a *Sprint Backlog*[6]
- **In Progress** - Tasks that are currently in progress. These tasks **must be assigned to whoever is working on them**
- **In Testing** - Where relevant, tasks move into this column if unit or integration testing is needed on that task (e.g., a new API endpoint should have unit tests written). **These tasks should be assigned to whoever is writing the tests**

[3] See `https://trello.com/b/0uh6AZdu/doubtfire-backlog`

[4] See `https://www.scrumalliance.org/articles/39-glossary-of-scrum-terms#1125`

[5] See `https://trello.com/b/lxTvHiQ9/general-development`

[6] See `https://www.scrumalliance.org/community/articles/2007/march/glossary-of-scrum-terms#1117`

for the task

- **In Internal Review** - Tasks that are to be reviewed internally by another team member. **These tasks should be reassigned to the reviewer**
- **In Open Pull Request** - Tasks that have been put into a Pull Request and assigned to a product owner for external code review. **These tasks should be reassigned to the external code reviewer**.
- **Done** - When the task is merged into the `develop` branch (the Pull Request has been closed).

3.3.1 Follow Up Actions

1. **Alex, Jake, Declan - By March 18** - Begin working on the SDLC plan based on the notes taken above
2. **Alex - ASAP** - Add the SDLC plan in a general context to the Doubtfire GitHub wiki for future Doubtfire developers and how they can work on a card (include Trello screenshots and the process)

---

4 Meeting 9 Mar at 4:30pm

Location: ATC Project Rooms

4.1 Present

- Alex Cummaudo
- Reuben Wilson
- Lachlan West
- Declan English

4.2 Agenda Items

1. Follow up action item 5 from Monday's meeting[7] for Declan and Lachlan on dual-booting Ubuntu

[7] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#follow-up-actions-1`

2. Alex to discuss new Trello board[8] layout and process
3. Team to decide which cards they want to work on for upcoming sprint
4. Alex to discuss git workflow process[9]

4.3 Discussion, Decisions and Agreements

4.3.1 RE: 1. Follow up action item 5 from Monday's meeting for Declan and Lachlan on dual-booting Ubuntu

- Lachlan is in the process of installing Ubuntu
- Declan is currently using Windows 10 for documentation, will install Ubuntu ASAP
- Alex to guide Doubtfire setup on Ubuntu machines when they are ready

4.3.2 RE: 2. Alex to discuss new Trello board layout and process

- Alex walked through Trello features, Doubtfire Backlog board, Helpdesk Ticketing board, including layout of boards and task completion separation
- Task size to be added to a card's title based on how long the assignee thinks it will take for them to work on it:
- `XS` = less than a day
- `S` = one day
- `M` = three days
- `L` = one working week
- `XL` = > 1 week
- E.g., `[XL] Task title name`
- Cards to be moved to appropriate columns based on state of task
- Assignees changed when moving between columns (especially when `In Review` and `In Open Pull Request`) to reflect status of task

[8] See `http://trello.com/doubtfire`

[9] See `https://github.com/doubtfire-lms/doubtfire-api/blob/develop/CONTRIBUTING.md#getting-started-with-the-workflow`

4.3.3 RE: 3. Team to decide which cards they want to work on for upcoming sprint

- Cards will be claimed by team members as they work on them, ensuring a single team member works on a single task at a time
- Basic rundown of existing bugs given by Alex during the meeting. Existing bugs are not expected to be large or high difficulty
- Tasks in the `Doubtfire Backlog` have priorities that **are not** respective of difficulty (e.g., low priority tasks may be difficult)

4.3.4 RE: 4. Alex to discuss git workflow process

- Extensive GitHub intro presented
- Team introduced to forking and branching, including standards and project conventions
- Project will be forked as `final-year-project` from Doubtfire development, then each task will be branched on the relevant fork off `develop`.
- Code quality will be ensured by:

1. running a quick code walkthrough when the card moves to `In Internal Review`
2. submitting a pull request from to the `final-year-project` fork to the `doubtfire-lms` repository

4.3.5 Additional Items

- Further information given on the setup of GitHub authorship and pulling. Included the following commands:

1. `git config --global`
2. `git config --user.name "<name>"`
3. `git config --user.email "<email>"`

- Alex will need to document the above in the `CONTRIBUTING.md` document
- Went through basic introduction to Java/CoffeeScript
- Went through setting global `.gitignore` files using gitignore.io[10] using:

[10]See `http://gitignore.io`

- `grunt`
- `rails`
- `ruby`
- `osx` (if applicable)
- `linux` (if applicable)
- `xcode` (if applicable)
- `intellij` (if applicable)

4.4 Agenda Items For Next Meeting

1. Get a status update from each team member on how they are progressing through their tasks

4.4.1 Follow Up Actions

1. **Declan, Lachlan - By March 14** - Install and configure Doubtfire and git on Ubuntu
2. **All - ASAP** - Look up Trello cards and pick and choose a few to get started on
3. **All - By end of Sprint 1- March 21** - Work on Trello tasks from (2)
4. **Alex - ASAP** - Document `git config` settings in `CONTRIBUTING.md`

---

5 Meeting 20 Mar at 1:30pm

Location: Skype

5.1 Present

- Alex Cummaudo
- Reuben Wilson
- Jake Renzella
- Lachlan West
- Declan English

5.2 Agenda Items

1. Retrospective on Sprint 1
2. Unit assessment
3. Upcoming project sprint

5.2.1 RE: 1. Retrospective on Sprint 1

- Declan unable to set up Ruby on Ubuntu at this stage. He will move to do more work in documentation and attempt to make Ruby work in the meantime.
- Jake worked on the web app:
- Adding the functionality to press the return key to submit a comment[11]
- Resizing the calendar text fields[12]
- Jake will continue by spiking backend unit testing with Rails with Reuben and share his findings with the team
- Reuben worked on the web app:
- Adding a label to the header dropdown of the current unit[13]
- Fixing duplicate student enrolments[14]
- Alex refactored the stylesheets to SASS[15]
- Alex helped Jake, Reuben on learn the codebase and has been helping Declan and Lachlan work through installation issues
- Lachlan will begin working on the add alt-key[16] task
- **Reflection:**
- While it was good that some team members got started on the tasks, it was disappointing that others could not
- The team needs to start working tighter and ensure that tasks are being done, whether slowly or not slowly
- In the event that you can't work, it should be made informed to the group
- It is expected that each team member works **at least half an hour a day** on Doubtfire, on average **8-10 hours a week**

[11] See `https://github.com/doubtfire-lms/doubtfire-web/pull/11`
[12] See `https://github.com/doubtfire-lms/doubtfire-web/pull/2`
[13] See `https://github.com/doubtfire-lms/doubtfire-web/pull/18`
[14] See `https://github.com/doubtfire-lms/doubtfire-web/pull/15`
[15] See `https://github.com/doubtfire-lms/doubtfire-web/pull/14`
[16] See `https://trello.com/c/QSAWPy36/35-add-in-alt-keys-using-accesskey-attribute-to-quickly-perform`

5.2.2 RE: 2. Unit assessment

- User manuals not suitable for Doubtfire LMS
- Training sessions run by the developers are suitable
- Adding *How Do I...* GIFs on the Doubtfire wiki would be better
- Team agreed that it's generally found most users of Doubtfire wouldn't read through lots of texts
- Documentation requirements for this to be discussed with Andrew Cain on 21 Monday meeting to confirm if user manuals are necessary
- SDLC plan documents and `CONTRIBUTING.md` documents are far more important than a requirements document—this will be contributed as an assessment document
- Especially since requirements are changing in the agile-managed Doubtfire, it may not be appropriate to have a static requirements document—we will discuss this with Andrew Cain and Graham Farrell
- Agreed that a final prototype presentation instead of a film—a live demo would be more suitable but if a film is required the team will devise one
- Discussed creating final mockups rather than prototypes, as prototypes will not function without being incorporated into the existing production-level Doubtfire system
- Agreed that the assessment should a **combination of documentation and product-based assessment**:
- e.g., a prototype iPad app and the backend functionality completed
- e.g., the wiki documentation of Doubtfire "how tos"
- Agreed to record changes as they are made through the use of Trello and `final-year-project` wiki, as the agile development of the project will make a solid initial requirements analysis very difficult
- Team will meet with the project client and supervisor to determine more accurate and reasonable metrics for marking as per Swinburne requirements

5.2.3 RE: 3. Upcoming project sprint

- A meeting with Andrew Cain determine the next phase of development is being held Monday 21 March
- In the meanwhile, or unless changed by Andrew Cain, the team will continue to work with a stronger emphasis into unit testing and back-end functionality

5.3 Agenda Items For Next Meeting

1. Get a status update from each team member on how they are progressing through their tasks
2. Discuss what will be achieved in the upcoming sprint
3. Agree on unit assessment guidelines with Andrew Cain and Graham Farrell
4. Devise next sprint with Andrew Cain

5.3.1 Follow Up Actions

1. **Declan, Lachlan - By Monday 21/3** - Devise agenda for assessment criteria meeting with Graham and Andrew
2. **Reuben, Jake - In two weeks** - Come back to the group regarding the spike on back-end testing framework. Organise meeting with Andrew Cain and the rest of the Doubtfire team to share spike results.
3. **Declan - ASAP** - Ensure Doubtfire is running or else work on documentation

---

6 Meeting 20 Mar at 1:15pm

Location: EN509a

6.1 Present

- Graham Farrell
- Alex Cummaudo
- Reuben Wilson
- Jake Renzella
- Lachlan West
- Declan English

6.2 Agenda Items

1. Requirements Analysis Deliverable
2. Assessment Criteria

3. Current Project State

6.3 Discussion, Decisions and Agreements

6.3.1 RE: 1. Requirements Analysis Deliverable

- Okay to use Trello as requirements documentation
- As the board keeps changing, it doesn't make sense to have a static document that outlines requirements that will change
- However, Graham wants something tangible—some form of document for this
- Graham is to be added to our Trello project board to see how we are progressing
- This will also depend upon Graham and Andrew meeting and discussing a suitable document

6.3.2 RE: 2. Assessment Criteria

- Graham follows a standard assessment criteria document
- However, we can talk to Andrew Cain and work out something that may better suit the uniqueness of this project

6.3.3 RE: 3. Current Project State

- Reuben, Jake and Lachlan are currently working through small bug-fixes or minor enhancements to help get start learning the Doubtfire codebase
- Jake and Reuben are working on preparing the minitest suit
- When nothing else to do, Jake will work on Doubtfire design of marking page.
- Declan is still setting up Doubtfire. He will need to get this fully resolved ASAP or at least work on documentation.
- Alex is managing the Trello boards and high-level documents regarding project management. He is also assisting all other team members with domain knowledge of Doubtfire as much as possible.

6.4 Agenda Items For Next Meeting

1. Current Project State
2. Review of Assessment Criteria

3. Review of Requirements Documentation

6.4.1 Follow Up Actions

1. **Alex** - **ASAP** - Add Graham to Trello Board

---

7 Meeting 20 Mar at 1:30pm

Location: EN513b

7.1 Present

- Andrew Cain
- Alex Cummaudo
- Reuben Wilson
- Jake Renzella
- Lachlan West
- Declan English

7.2 Agenda Items

1. Requirements Analysis Deliverable
2. Assessment Criteria
3. Upcoming Sprint

7.3 Discussion, Decisions and Agreements

7.3.1 RE: 1. Requirements Analysis Deliverable

- Andrew wants some sort of high-level, conceptual requirements document
- The purpose of this is to ensure that Andrew and the team both understand and agree upon the design of the system
- Very limited amounts of text, just diagrams that illustrate the work to be done on both the back and and front end(s) (there may be multiple front ends, e.g. iPhone/iPad/Android/Web)

- Architecture diagrams should be basic (not UML)—just highlight how the back end and front end will communicate
- Requirements documentation can also include prototype images of iOS app, prototypes of the web and overall architecture documents with diagrams on how it could work (conceptual design)
- Use keynote as a design tool to create basic prototypes
- Discussed using a redis queue[17] as an object persistence pipeline. Similar abstraction level as the database but redis is a queue and memory management pipeline
- The redis queue will sit behind the API (i.e., no direct communication with the redis queue)
- Use of Twitter Analytics Framework (Fabric) to replace Google Analytics
- Performs better at real-time analytics
- Just replace the provider in the `analyticsService` in the web app
- Will do a lot of the graph work for us
- Use live data to create interesting graphs.

7.3.2 RE: 2. Assessment Criteria

- Refer to the final year project unit learning outcomes—adapt these into the Assessment Criteria
- Don't make it a product-based criteria as this gives too much pressure on what to deliver (i.e., need 3 new apps and a backend—too much and would be a shame to lose a grade even when we tried our hardest)
- Should be related to what you have done related to your role in project
- P is you got DF installed and worked on at least one or two tasks
- C is you actually worked on more than what was required
- D is you had significant contribution in a single area
- HD is you showed some initiative and had significant contribution in multiple areas
- To be worked out further with the group in a future meeting
- Examples of roles:
- Jake and Reuben for testing
    - how well they relay their findings/teach to the rest of the group

[17] See `http://redis.io`

  - how many bugs they were able to find with their testing
  - how well they covered testing from almost 0% to what they come up with

- Alex for team lead

  - leadership skills
  - helping people (not just giving orders)
  - assisting with Doubtfire domain and codebase knowledge
  - managing the Trello board and team

7.3.3 RE: 3. Upcoming Sprint

- Assign two weeks for this sprint (March 20 to April 4)
- Choose some new tasks for working on from the product backlog
- Also start working on requirements documents and assessment criteria for our project **concurrently**
- Jake and Reuben to start on unit testing of backend using minutest
- Be mindful of adding task sizes[18] as this wasn't done in last sprint[19]
- How long do you take the task to expect and how long it actually took [S/L] (expected task sizes vs actual task sizes (use a slash to distinguish))
- Don't worry about the slow velocity at the moment, it is expected that when learning a new system it will take a while to start completing tasks
- Would be good to compare estimated task sizes to actual task sizes especially between the start and end of the year to show how much you have improved (relate back to Agenda Item 2)
- Skype meeting this weekend regarding organisation and documentation

7.4 Agenda Items For Next Meeting

1. Choose some more tasks from product backlog
2. Agree on Assessment Criteria
3. Start working on Requirements Documentation

[18] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#re-2-alex-to-discuss-new-trello-board-layout-and-process`

[19] See `https://github.com/final-year-project/documentation/wiki/Meeting-Minutes#re-1-retrospective-on-sprint-1`

7.4.1 Follow Up Actions

1. **Alex** - **ASAP** - Organise a Skype meeting with everyone for this weekend

---

8 Meeting 28 Mar at 1:30pm

Location: Skype

8.1 Present

- Alex
- Jake
- Reuben
- Lachlan

8.2 Agenda Items

1. Writing up assessment criteria.
2. Requirements Documentation & Diagrams
3. General sprint information with cards and progression.

8.3 Discussion, Decisions and Agreements

8.3.1 RE: 1. Assessment Criteria

- "Role based" assessment criteria.
- Google docs to start developing the document.
- We are all going to contribute to document.
- We all agree on this method.
- Will take this document to Graham.
- Define and breakdown of roles and requirements for roles.
- Will use Learning Summary Report to justify grades.

8.3.2 RE: 2. Requirements Documentation & Diagrams

- Lucid Chart for rough conceptual diagram for Redis.
- Alex is knowledgable with Redis so will start document.

8.3.3 RE: 3. General sprint information with cards and progression

- Alex will start looking at some of his sprint inbox items.
- Screenshots on Trello cards are very important.
- Important to hassle each other when dealing with tasks.
- Sizing of cards is important, don't forget.
- After finishing a card, also add how long the card actually took.

8.4 Agenda Items For Next Meeting

1. Development session to discuss cards in person.

8.4.1 Follow Up Actions

1. **Team - April 4th** - Delivering assessment documents to Graham (requirements, assessment, LSR).
2. **Jake and Reuben - When ready** - Update Doubtfire API wiki to include minitest documentation
3. **Jake and Alex - Week of April 5th** - Meet to discuss app designs
4. **Jake and Reuben - Week of April 5th** - Meet with Andrew to finalise minitest requirements.
5. **Lachlan - Week of April 5th** - Work on his two backlog cards.
6. **Team - Wednesday 6th of April** - Book room for development session (AMDC 401).

---

9 Meeting 18 April at 1:30pm

Location: Andrew's Office

9.1 Present

- Alex
- Jake
- Reuben
- Lachlan
- Andrew Cain

9.2 Agenda Items

1. Feedback from iOS and Android prototypes
2. Feedback for architecture of Helpdesk API

9.3 Discussion, Decisions and Agreements

9.3.1 RE: 1. Feedback from iOS and Android prototypes

- Andrew is unsure about a single queue
- He would like tutors to have their own queues, and they can take as many students as they want from the **global queue**
- Whenever students submit tickets, they will be pushed into the global queue
- The idea is to keep the global queue as empty as possible; tutors take students off the global queue and run through their own queue
- The global queue is sorted by time ticket was submitted, grouped by subjects the tutor teaches and subjects the tutors do not teach
- The tutor queue is sorted by time ticket was last 'reviewed' by a tutor
- A tutor can then remove the ticket from their queue once the ticket is marked as complete by the tutor
- Tutors can refer students from the global queue to *other* tutors if need be
- Introduces minor complexity to the app but makes it run more efficiently as one tutor will deal with a group of students at a single time

9.3.2 RE: 2. Feedback for architecture of Helpdesk API

- Andrew is happy with the idea but confidentiality is a problem.
- If we were to host on Heroku, we would only store Doubtfire Rails IDs on Redis

- Andrew is a bit worried about reporting; we would need to investigate
- Andrew is worrisome about making staff use Google Analytics; will need to investigate at a later stage

9.4 Agenda Items For Next Meeting

1. Show Andrew Revised Prototypes

9.4.1 Follow Up Actions

1. **Alex, Jake** - ASAP - Work on a revised prototype

---

10 Meeting 2 May at 1:30pm

Location: Graham's Office

10.1 Present

- Alex
- Jake
- Reuben
- Lachlan
- Graham Farrell

10.2 Agenda Items

1. Discussing current status of project
2. Discussing peer reviews

10.3 Discussion, Decisions and Agreements

10.3.1 RE: 1. Discussing current status of project

- Emphasis at this stage is to get the video complete
- Graham is happy with assessment criteria and requirements documentation

- Graham is concerned about state of some group members

10.3.2 RE: 2. Discussing peer reviews

- Happy that peer reviews were submitted and has given him some insight into the group
- Declan has resigned from the team

10.4 Agenda Items For Next Meeting

N/A

10.4.1 Follow Up Actions

N/A

---

11 Meeting 25 May at 1:15pm

Location: Graham's Office

11.1 Present

- Alex
- Reuben
- Lachlan
- Graham Farrell

11.2 Agenda Items

1. Discussing current status of project
2. Discussing submission protocol

11.3 Discussion, Decisions and Agreements

11.3.1 RE: 1. Discussing current status of project

- Discussed some of the health issues we have been having in the group
- Have all agreed that no one is to blame when it comes to health; it happens and we work together to resolve it
- Graham has emphasised that we have all done a good job this semester which is good motivation for the team

11.3.2 RE: 2. Discussing submission protocol

- Graham wants something tangible — we need to documentary the Wiki and print it for him
- Graham has stressed that our current requirements are lacking substantially. We must work together to submit:
  - Extended stakeholders
  - Extended goals and objectives
  - Extended risk analysis (missing altogether); this is a hurdle!
- We need to pull together and try and focus on producing this by next week
- Arrange time for next week for submission

11.4 Agenda Items For Next Meeting

1. Submit portfolio

11.4.1 Follow Up Actions

1. **Team** - ASAP - Work on fleshing out an SRS in more detail
2. **Alex** - By the weekend - Email Graham for a time we can submit the portfolio

Image Attributions

'Scrum Development Team' designed by Roy Verhaag

'Storage' designed by Dirk-Pieter van Walsum

'Technical Requirements' designed by Alvarobueno

'Quality Assurance' designed by Lorenzo Stella

'Agile' designed by Nimal Raj

'Concept Prototype' designed by Yu Luck

'LOG File' designed by Viktor Vorobyev

'Meeting' designed by Marie Van den Broeck

'Plan' designed by Michael Finney

'Profile User Flow' designed by Sharon Showalter

From The Noun Project collection

http://www.nounproject.com