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

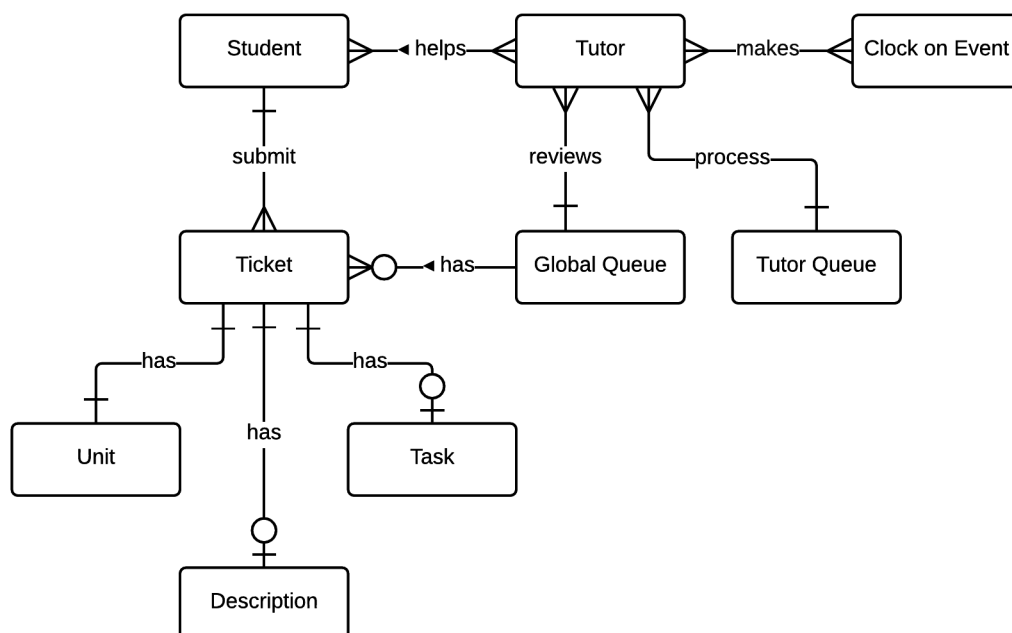Refer to the extended planning documentation for more on this.

2   Entities

Figure 1: Entity Relationship Diagram highlighting fundamental entities

Figure **??** 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 **??**.

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**
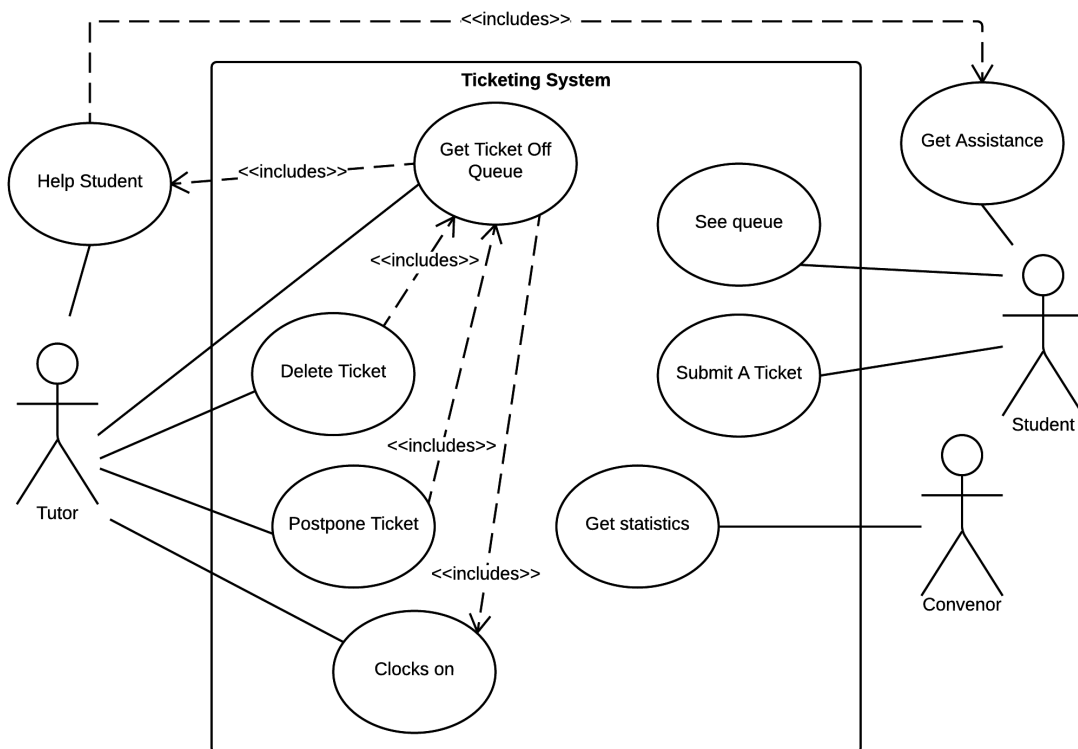
Step 5. Student views an estimate of wait time

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 **??**. 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 **??**.

Step 3. Tutor accepts the push notification and they are clocked on. See Figure **??**.

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 **??**.

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.

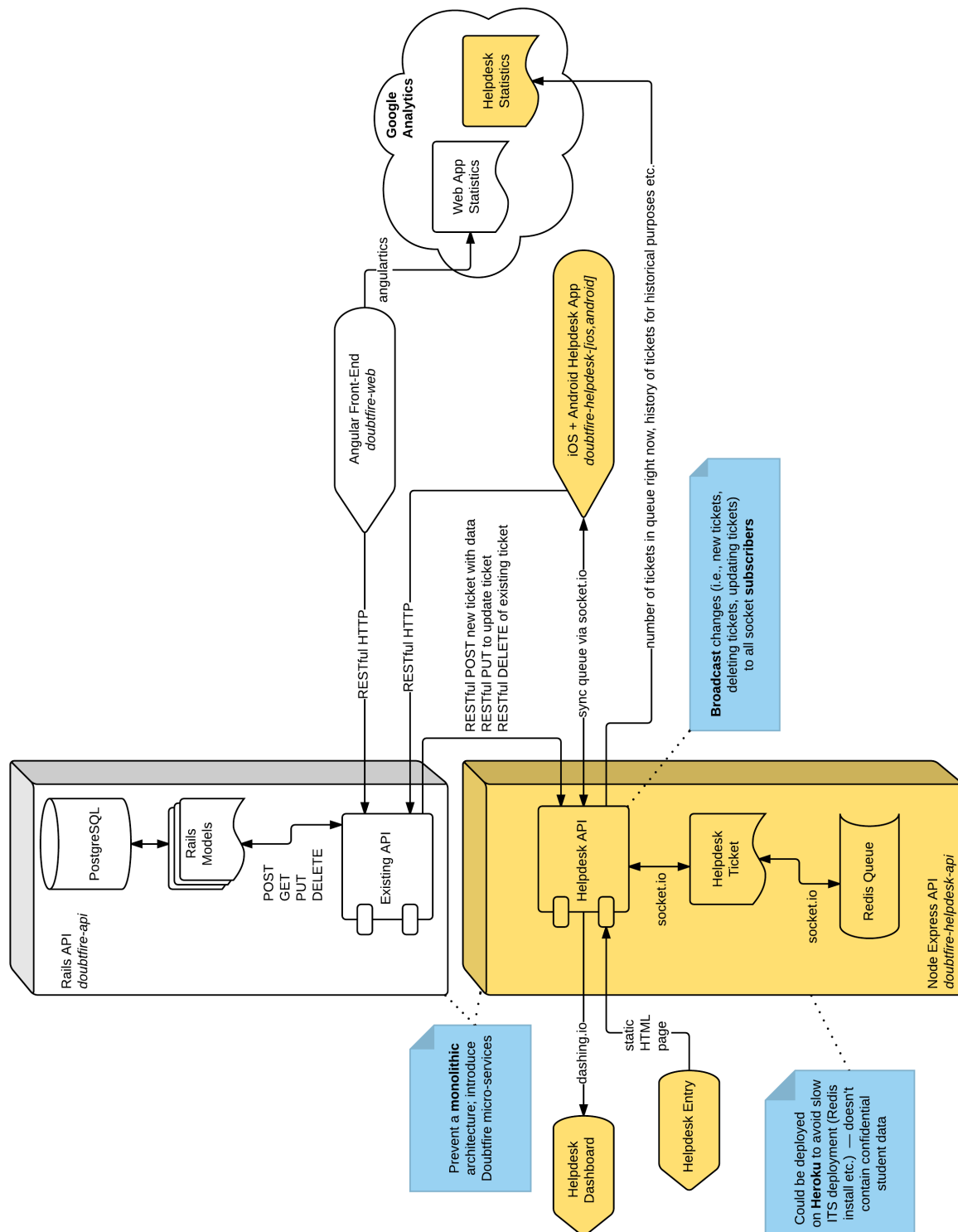Refer to Figure **??** for more.

Figure 3: High Level Architecture Diagram
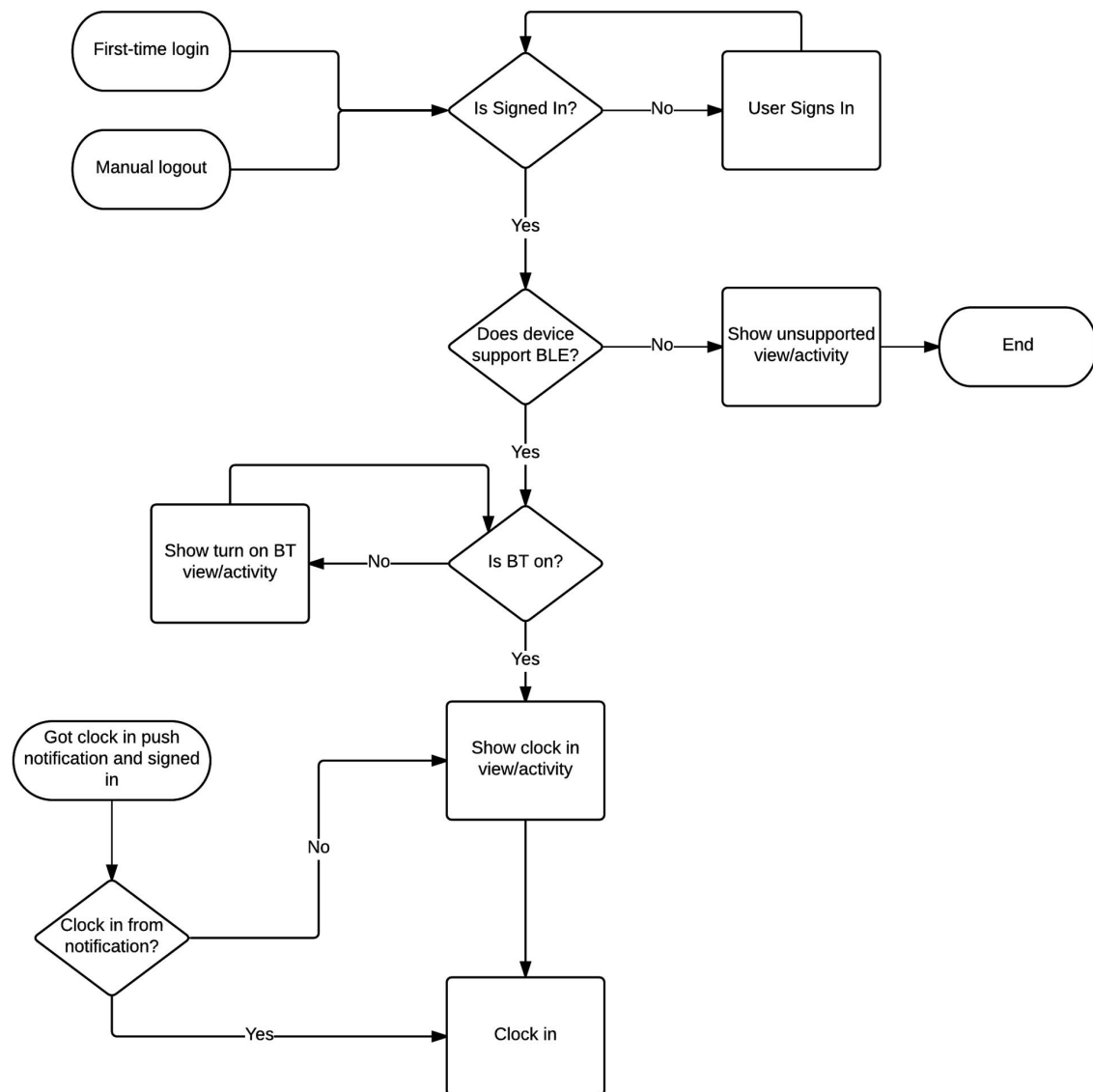
A   Supporting Use Case Images

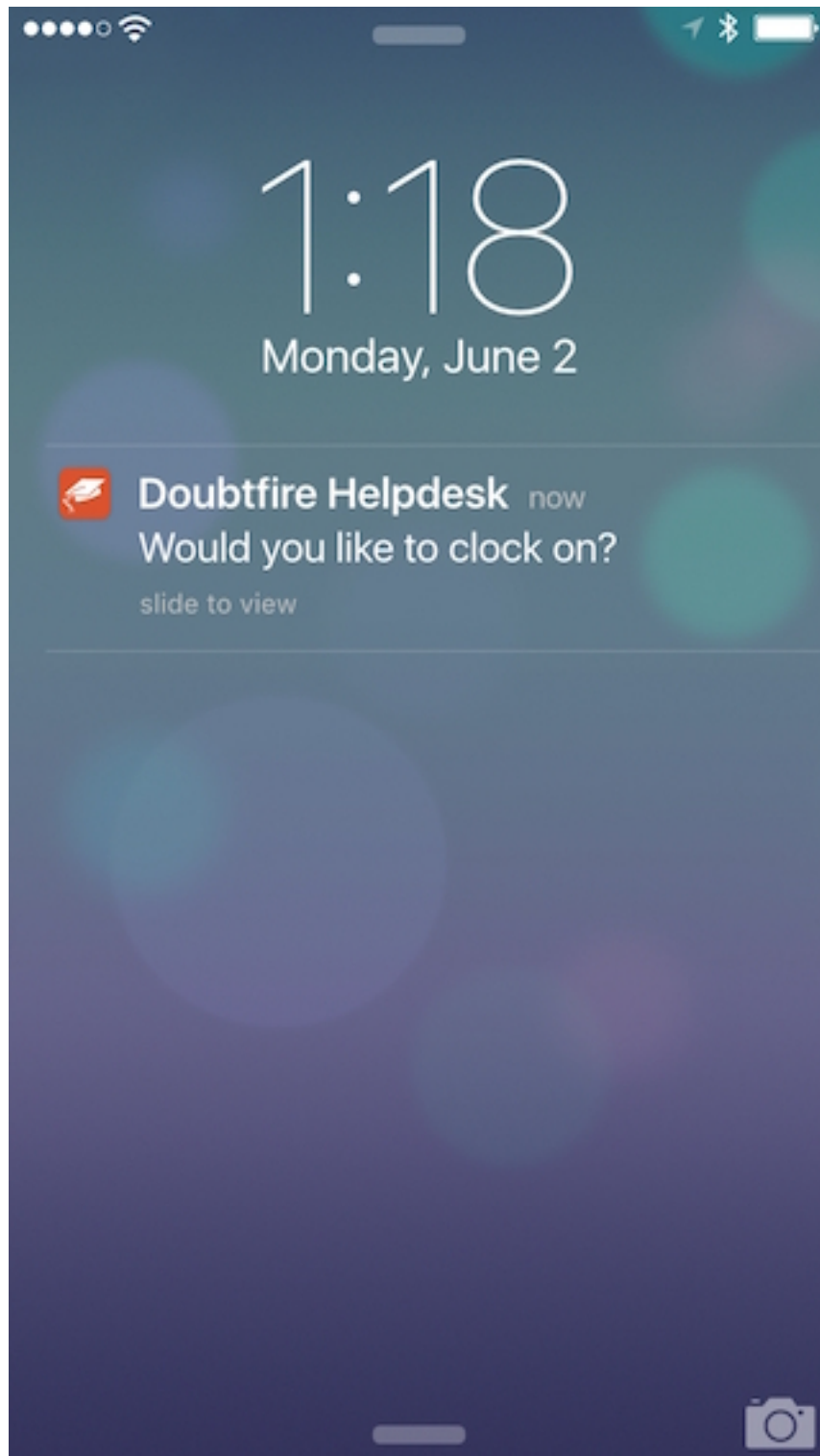Figure 4: Workflow diagram for clocking on
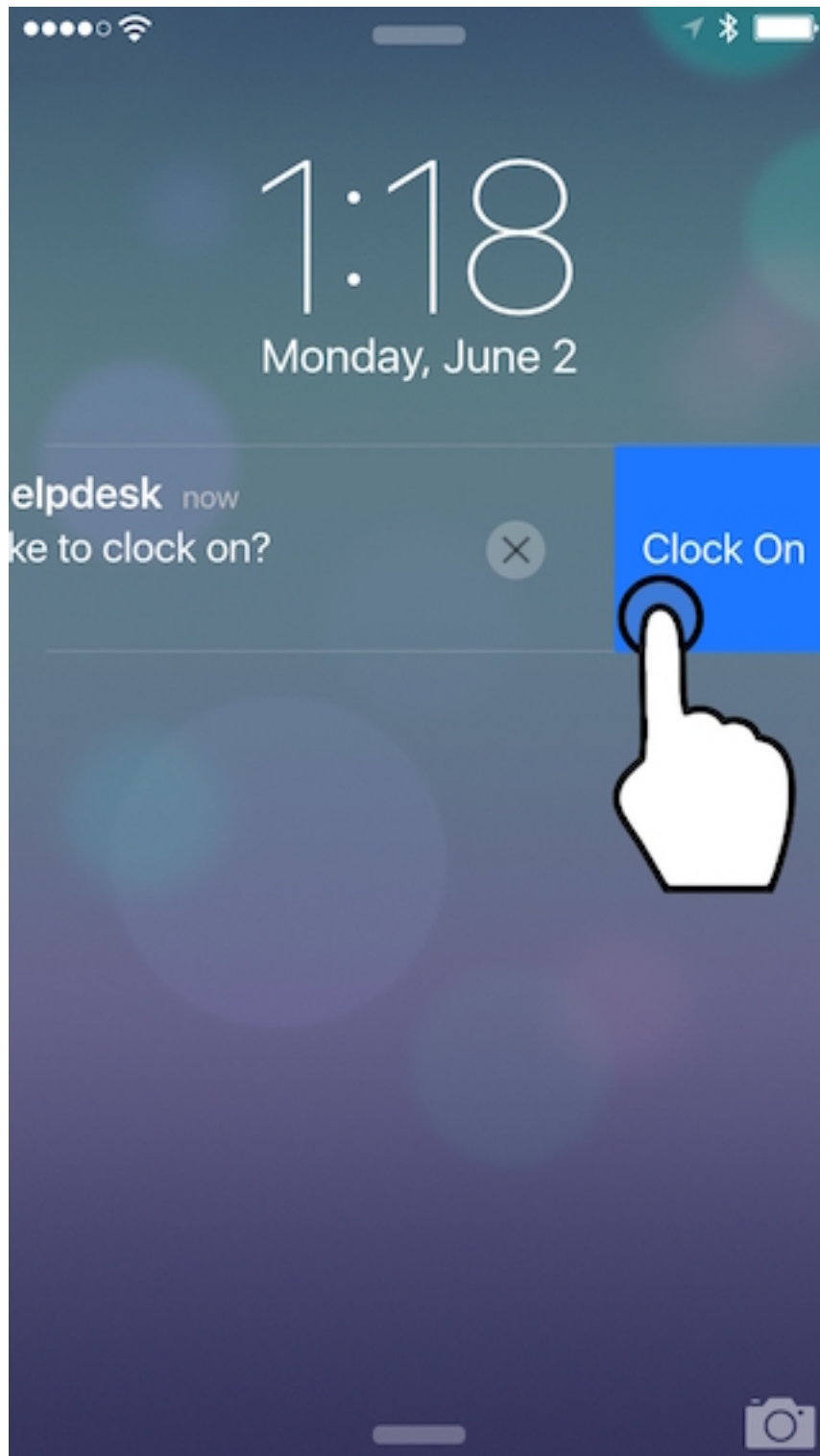
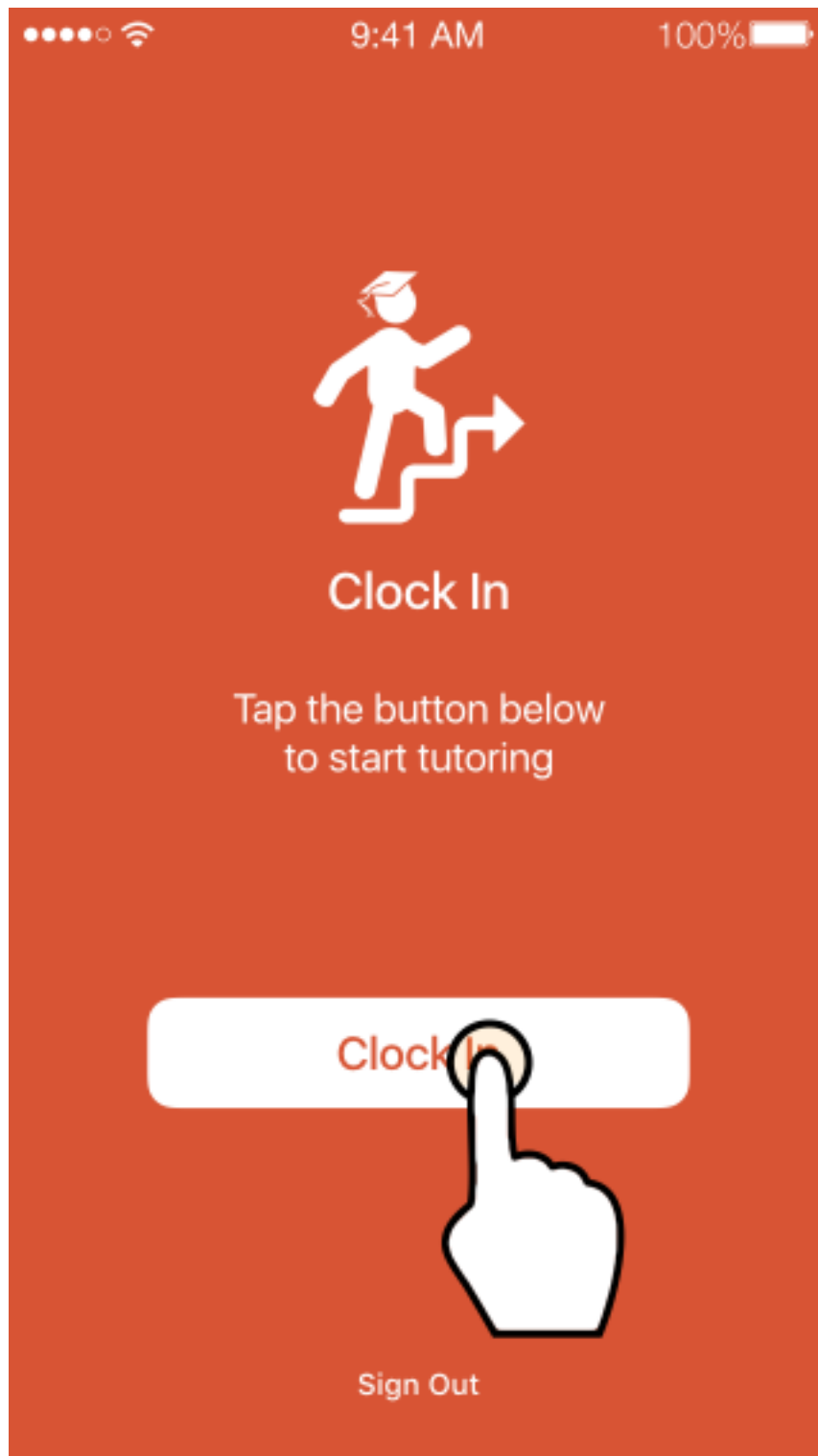Figure 5: Push notification to clock on

Figure 6: Clock on accept via notification

Figure 7: Clock on manually