How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: “**Capstone_Stage1**”
3. Replace the text in green

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”

GitHub Username: dev-sobo

TimeCard

Description

Write a brief summary of what your app does. What problem does your app solve?

This app solves the classic problem of mismatching work time information. TimeCard allows you to keep track of how much time you should be paid for at your place of employment, allowing you to deduct time for legally-required lunch times, as well as allowing you control special OT rules, rounding to scheduled work times, absent or late shifts, and paid-time-off.

Intended User

My intended user are those hourly employees who seem to have “paycheck anxiety” every week, constantly needing to check their pay stubs to see if the hours that they have worked are actually being paid for. This takes all the manual calculation of hours out of the equation, providing a simple user interface for clocking in and out of a shift, and a bird's-eye view of their shifts in a calendar.

Features

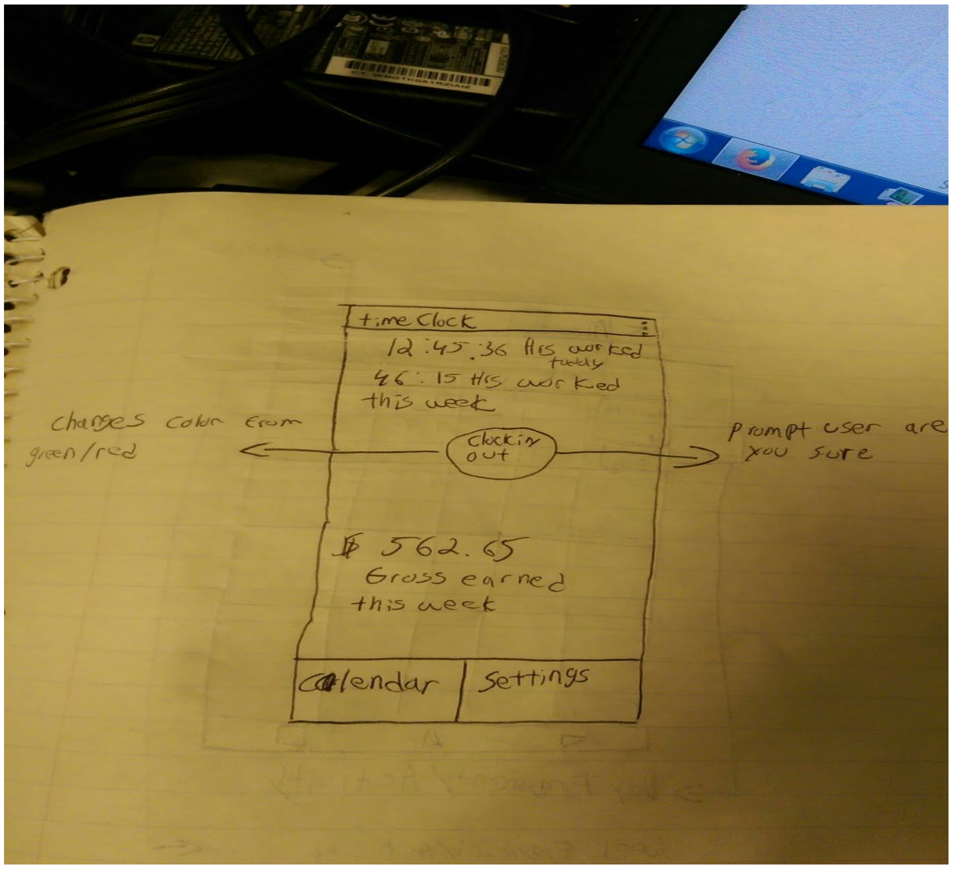
List the main features of your app.

- Keeps track of time worked on a shift per shift basis
- Shifts can have special qualities to them i.e. OT qualified or PTO.
- Hours worked per week & shift are kept track of
- Gross earnings per week & shift are kept track of
- Easy to use interface to clock in and out
- Calendar generated for shifts allows for easy editing of shifts, such as in time and out time.
- Option to have a shift-differential, if after and before a certain time in a day, pay can be adjusted.
- Option to set after x amount of hours a break is automatically taken out of a shift’s gross pay calculation.

User Interface Mocks

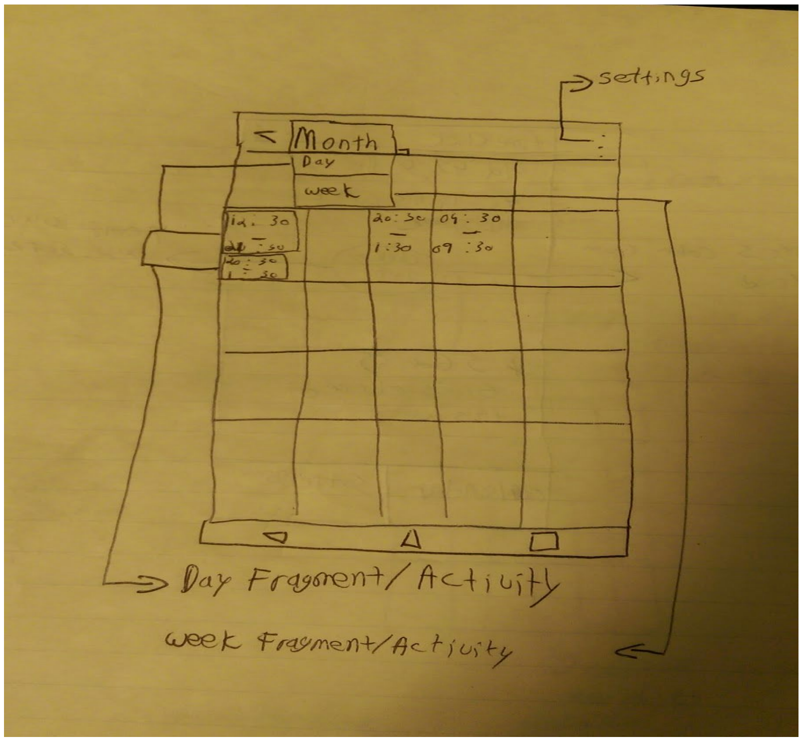
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Clock-in, Clock-out - Main Activity


time Clock
12:45:36 Hrs worked today
46:15 Hrs worked this week
changes color from green/red
Clock in out
Prompt user are you sure
$562.65 Gross earned this week
Calendar
Settings

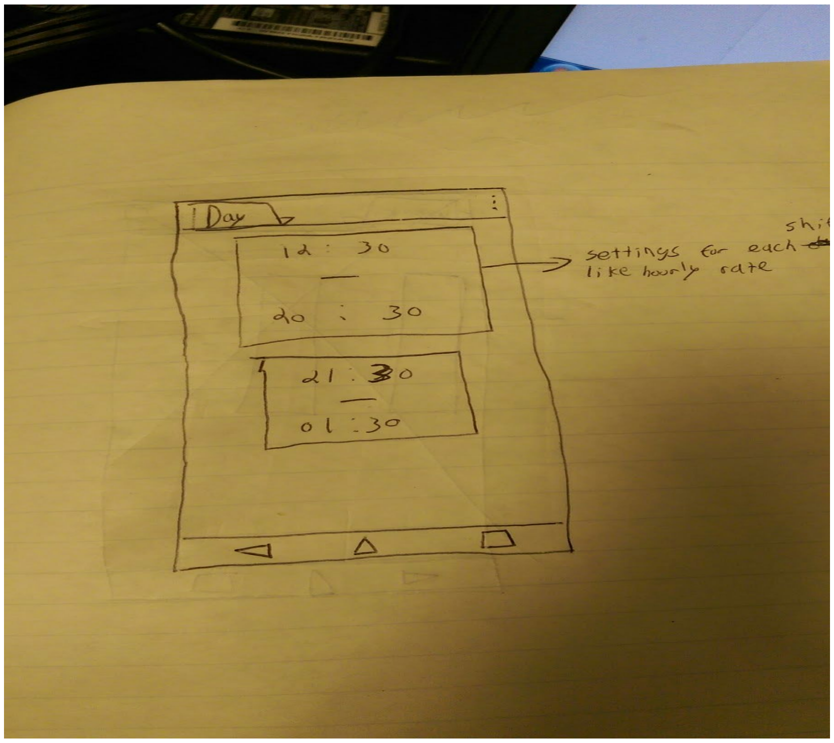
This is the first, and main screen of the app. This is where you can clock in, and then clock out for a shift. It shows how many hours you have worked in a day, and in the week, plus what your gross pay should be. It also provides navigation to other parts of the app, the Calendar view and settings. In the final version of the app, there will be a navigation drawer available.

Calendar - Month View/Activity



This is the Calendar view, showing the entire month. Although it's hard to see here, it shows each shift, from start time to end time. Changing the month, day, and week view is handled by a spinner. Clicking on an individual shift allows the user to edit the shift's start and end times.

Calendar - Day View - This is a *possible* feature.



This is the day view that was mentioned above, available via spinner or click on an individual day. Click on a shift gives you options for that individual shift, for example if you have a different pay-rate after a certain time, if you were late to the shift and need to adjust how much time you really are paid for, or if you were absent for the shift.

Here is another mock-up I made on the internet:

https://www.fluidui.com/editor/live/preview/p_EaPLOFpztfISdoduFT42tJyGoWsGpQrT.1459923633895

Key Considerations

How will your app handle data persistence?

Describe how your app with handle data. (For example, will you build a Content Provider or connect to an existing one?)

This app will be using a new content provider, with it's data backed by a SQLite database. This data will be a simple one-table database, describing an individual shift for each row. It's attributes are:

- **Start Time**
- **End Time**

- **Hourly Pay**
- **Day of the week**
- **Day of the Month**
- **Month name/number**
- **Year**
- **Number of hours for each shift**
- **Gross pay for each shift**

A calendar, using either the Caldroid library, or square's 'Time Square' calendar, will be populated with the user's shifts. I found a good The weekly gross pay will be calculated as well.

Describe any corner cases in the UX.

I would like to eventually implement the ability to add your whole schedule at once, so you can see what you need to work ahead of time, plus being able to edit the start and end times of your shifts as you see fit.
This would need to include:
Paid Time off days, Absent days (including possible Occurrence/points tracking), as well as the ability to put in "theoretical" shifts in.
Ability to edit when, if any, breaks are taken out based on local labor laws past a certain number of hours worked. -> Could possibly alert the user if they are coming up to a time where a break will be taken out of their pay via a notification.

Describe any libraries you'll be using and share your reasoning for including them.

Schematic to make a content provider
Caldroid for the calendar
Caldroid itself uses a useful library for date/time handling and conversions, known as DATE4J.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

If it helps, imagine you are describing these tasks to a friend who wants to follow along and build this app with you.
I will use the Schematic and Caldroid libraries to handle Content Provider generation, and providing a User Interface for a calender, respectively..

- Set up Schematic like this:

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'

buildscript {
  repositories {
    mavenCentral()
  }

  dependencies {
    classpath 'com.android.tools.build:gradle:{latest-version}'
    classpath 'com.neenbedankt.gradle.plugins:android-apt:{latest-version}'
  }
}

dependencies {
  apt 'net.simonvt.schematic:schematic-compiler:{latest-version}'
  compile 'net.simonvt.schematic:schematic:{latest-version}'
}
```

- Set up Caldroid like this:
  Add ‘compile 'com.roomorama:caldroid:3.0.1’ to build.gradle file.

Task 2: Implement Utility classes and functions/content provider:

- Classes/Functions needed for: - A separate Utility class may be needed
  - Database interaction and management - CRUD Functions for shift data. This will likely be the content provider’s responsibility.
  - Time tracking and counting for each shift - Starting time & Date, ending time & date
  - Calculating gross pay with user-supplied hourly wage while factoring in breaks after a certain number of hours worked
  - Handling dates, including correctly splitting up hours worked each day into discrete shifts, including shifts that overlap from day to day. Split these up into hours worked for each week, and those weeks into months.
  - Clocking-in should have date and time stored into a database. UNIX time is useful here to calculate total amount of hours worked for each shift.
  - Clocking-out will have ending date and times stored into the same above record. Subtract Ending UNIX time from starting UNIX time to get total amount of time worked. Multiply this by the user-supplied hourly wage.
- Implement a content provider for the database of shifts

Task 3: Pulling data from RosterApps:

- Using jSoup library to login and pull data, it is possible to compare and contrast the differences between what the user is clocking in and clocking out, versus what is pulled from what they are scheduled for.
- This can be accomplished by a SyncAdapter, which will pull data from RosterApps periodically, and populate the user’s calendar with the pulled data.

Task 4: Implement UI for Each Activity and Fragment

Main Activity Tasks:

- Present the user with a summary of shift information which includes:

Current or Last-Clocked-In shift hours worked, and gross pay and break taken out, if any.

Current Week worked, shift hours and gross pay.
Number of Points/Occurrences accrued, this will be self-reported.
Button for clocking in and out of a shift
Navigation drawer for switching between the various activities.

- Implement an app bar for navigation and information.
- When the user first starts the app, ask for their hourly rate
- When a worker clocks in, record the following data to the database:
  - Start Time
  - Hourly Pay
  - Day of week
  - Day of Month
  - Month Name/Number
  - Year
- When a worker clocks out, record the following data to the same row as above for that shift:
  - End Time
  - Number of hours worked
  - Gross pay for shift.

Settings Activity Tasks:

- Options to change, probably stored as SharedPreferences.:
  - Hourly Rate
  - Start Day of Week

Calendar Activity (Month View) Tasks:

- Present the user with a month-view of their shifts that they have worked in the past.
- Grab the shift data from the database, and populate the calendar with the data in order to correctly place each shift on the calendar according to shift start & end time. Place gross-earned cash on each shift.
- Tapping on a shift allows the user to edit the clock-in and clock-out times with a pop-up dialog. Upon accepting the changes, the changes are updated in the database.
- Display ads at bottom of screen
- ***POSSIBLE EXTRA FEATURE:*** Each day and week should have a fragment associated with it to show a close-up view of those time periods. Should switch to these views via a spinner at the top of the month view. Ads can be shown when switching between views.
- Populate the calendar view with the pulled shifts from Rosterapps, and have the user-reported shifts overlayed on top of them, in order to see if they match up or not.

Task 5: Implement Google Play Services and Testing

- Implement Google Admob for both Main Activity and Calendar Activity.
- Implement Google Analytics to track user habits and what type of devices are using the app
- Test if shifts are properly being clocked in and clocked out and calculated

Task 6: Accessibility and Localization

Implement Accessibility and Localization features:

- Accessibility. Implement D-pad navigation as well as content descriptions.
- Implement a widget for clocking in and clocking out. If possible, create a widget with the calendar view.
- Ensure RTL layout is working, with all strings contained within strings.xml

Task 7: Building

Ensure building with Gradle works properly

- Implement correct signing and keystore configuration
- Ensure app builds and deploys using the installRelease Gradle task

Add as many tasks as you need to complete your app.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"