I Commander

4/11/2019
Version 1.0 - Black
Kier Lindsay, Samer Sefrani, Albert Lam, Sheil Mehta, Tianyang Zhou
https://github.com/syonfox/I_Comander
https://i-commander.herokuapp.com/
Dev Users are
joebob:asd – guest
starfox:asd – user(pilot)
cat:asd – admin
root:asd - superadmin

Overview:

I Commander is a mobile friendly web application which meets the following key functions. These functions are currently tracked manually in a spreadsheet and cumbersome for both the pilots and managers to maintain compliance with Canadian drone laws and regulations as well as company procedures. I Commander will assist drone pilots in following standard operating procedures (SOPs) by leading the user through a series of pre and post flight checklists. I commander will also track flight times – when the user completes the pre-flight checklist the app displays a "START Flying" button. Once Checked the app is in "active flight mode" and time is recorded until the user selects the "STOP Flying" option. The flight time is recorded in the parent system by user once the user is in wifi/data range.
I commander will send alerts/triggers actions – After the user has "stopped flying" they complete a series of Post Flight checklists. Items fall into several categories which trigger alerts and actions. For SOP compliance no alert is needed but pilots must acknowledge it. For Standard maintenance i commander will create a service tickets in the parent system service que based on post flight checklist items and details. For example, if the "underbody scratches and blemishes" item is checked a service ticket is created for that Drone. Some checklist items trigger a LOCK OUT event. This means the drone being used (the one for which the post flight checklist is being completed for) enters a "LOCKED OUT" Status. A service ticket is created and the drone is EXCLUDED from the Drone selection list meaning it can not be flown again until the service ticket is completed.

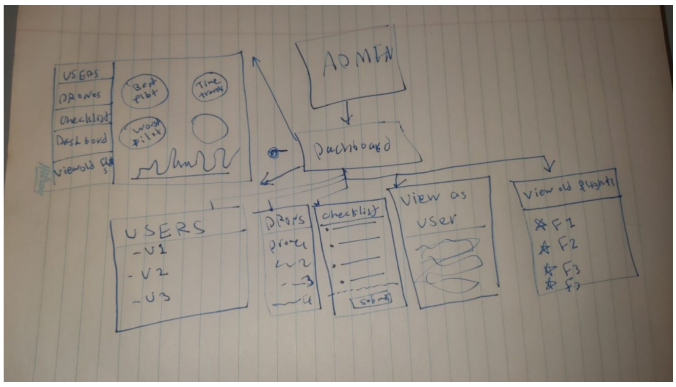
Competitive Analysis

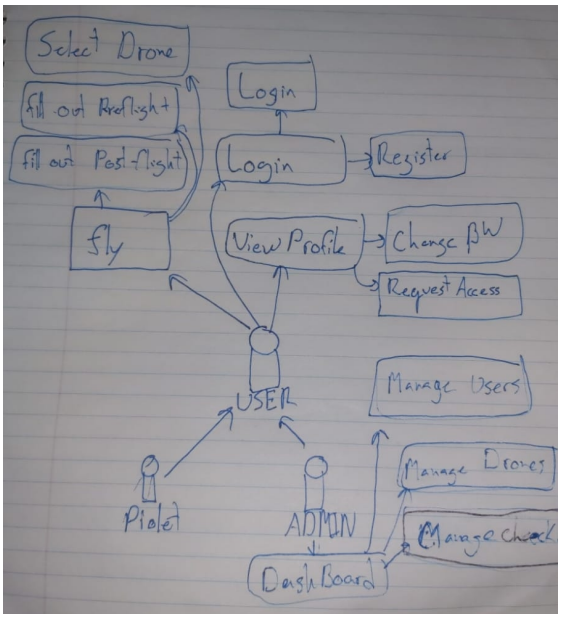
There are some other check list software out there but many of them are missing some of the key features required by the client such as:

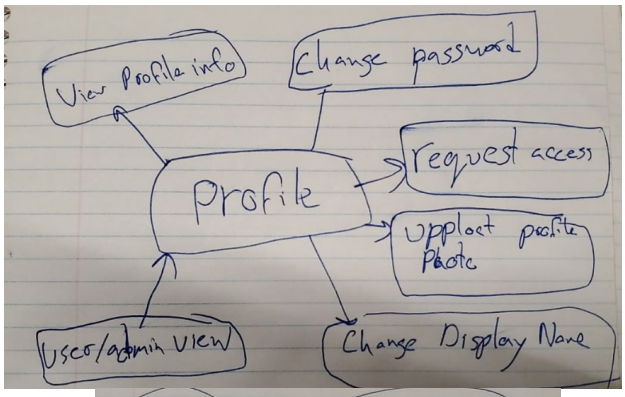
- ability to work offline and sync
- Not cross platform
- will not perform required action after being filled out.

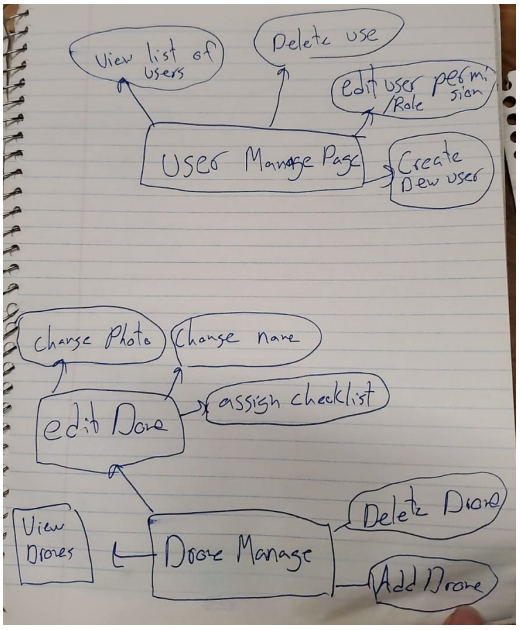
Our advantage is that we are specifically designing our app for use with drones to meet the many specialized requirements of the industry.

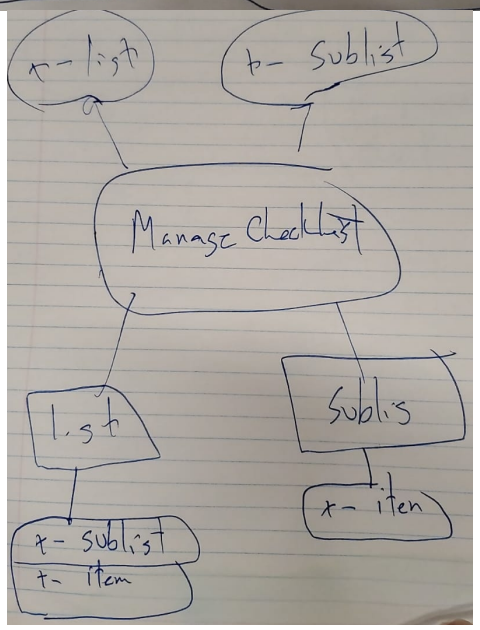
User Interface Requirements











User Stories

Actors:

Admin – Admins are the managers able to modify the systems and edit users. They also view the recorded flights.

- manage check lists
- manage users
- manage drones
- View flights
- simulate being a user to test configurations

User – Users are the pilots who will fly the drones they must be able to fill out the preflight checklists and submit them even when offline.

- Fly
    1. select drone
    2. fill out preflight checklists
    3. FLY :)
    4. fill out post flight checklist/ submit tickets if there are issues
    5. submit flight for review
- Check weather
- view SOP’s
- view profile
    - change password
    - request access to fly/admin
    - see stats

- login
    - login
    - register

I Commander will have two modes, Administration and Configuration Mode(admin mode) which is intended to be used by managers and available at the desktop level or inherited from the parent system. This mode will have features to manage users, create new users and manage their permissions It will also have a Drone management interface where the manager can add new drones as well as configure which users can access them and can assign checklists designed in the checklist module to the drone. The last core management mode will be the checklist management interface. This will allow the manager to design and edit modular checklists. It will also have the ability to "thread" checklist together making larger checklist specific to different drone types. Some checklists need to be filled in for ALL missions while others are drone specific. Admin user needs to be able to create new checklists selecting from several preconfigured field "types" and setting properties related to the "alerts" mentioned above (No action, Maintenance, LOCK OUT Maintenance), create sequences of checklists and assign checklist sequences to Drone models. The management mode will also have support for pdf SOP’s to be uploaded these can be limited to drones or viewed generally by the users.
A sample user story for this mode might be:

- Admin Goes to website
- Admin Logs in
- Admin Adds a new drone at the drone page
- Admin then makes custom checklist for this drone and added it to one of their previous templates
- Admin assigned this list to the drone
- Admin publishes the drone for users to use

The second major operation mode will be the Mobile User Mode. This will be the page available in the field for managing missions. The mobile mode will use web service workers and offline databases to work even when servers is unavailable or poor. This mode will have a simple interface and allow users to view company SOP’s or start new missions. When starting a mission it will flow through the stander operation first asking the user to select the drone they are flying then. It will prompt them to fill out the pre flight checklist before notifying they are ready to fly. Finally when they finish flying it will run through the defined post flight checklist and if there is any issues, the user notes problems with the drone and a maintenance ticket will be created. This system will also track the time from when a mission is started to finished. A sample user story for this would be:

- User logs in
- User creates a new Mission
- User selects the Drone from a list of available drones (Drones show as In Service or Locked Out)
- User completes pre-flight checklists
- User starts flight
- User stops flight
- User completes Post Flight Checklist
- New Mission

Another feature of the mobile mode might be gear checklists which are completed prior to leaving base or the field as opposed to mission specific checklists so we will design with this add on in mind.

User login

Name : cat

Actors : user
Triggers: user want to fly so they must login
System state: User not authenticated
Actions : prompt user to login, validate password, send response
post condition: show user view (Flight page), User authenticated
Acceptance tests: Check if user is authenticated
Iteration 1

User Signup

Name : cat
Actors : user
Triggers: user wants login
System state: User not authenticated, user does not exists
Actions : prompt user to sign up, check if username doesn't already exist, add user to db, prompt signin
post condition: user is in database, user sees login page
Acceptance tests: valid user is created and can login
Iteration 1

User Selects drones

Name : starfox
Actors : user
Triggers: User starts flight
System state: User authenticated, user in flight view
Actions : prompt user to select drone
post condition: drone is selected and appropriate checklists are shown
Acceptance tests: checklist page displayed for drone.
Iteration 1

User fills out checklist

Name : root
Actors : user
Triggers: User select the drone from drones page
System state: User is in before flight mode, User authenticated, user has selected a drone
Actions : Load checklist, user fill out a checklist, user submits a checklist
post condition: the flight starts and time is being recorded
Acceptance tests: current flight page displayed for drone. Checklist is saved into the database
Iteration 1

View and Edit checklists

Name : root

Actors : admin
Triggers: admin wants to create a checklist for a new drone
System state: admin is logged in
Actions : show existing checklists from database, add new item to checklist, save item to checklist,
post condition: updated checklist, checklist added to the database
Acceptance tests: checklist is accessible
Iteration 1

View users

Name : dog
Actors : admin
Triggers: admin wants to view existing users in the database
System state: admin is logged in
Actions : show existing users from database
post condition: admin sees the list of users
Acceptance tests: new user can be seen in the list of users
Iteration 1

Granting/removing permissions from users

Name : bobby
Actors : admin
Triggers: admin wants to give/take away permissions to users to make them(pilot/admin)
System state: admin is logged in
Actions : show existing users from database, admin assign a new role to a user
post condition: user role is updated in the db
Acceptance tests: user has updated permissions
Iteration 2

Submitting a Flight

Name : bobby2
Actors : user
Triggers: user finishes his flight
System state: user is logged in,user is flying
Actions : submit flight data to db
post condition: flight data is in the db, pilot will see a post flight checklist
Acceptance tests: flight is added to the db
Iteration 2

Viewing their own profile

Name : bobby3
Actors : user, admin
Triggers: user wants to view their own profile
System state: user is logged in, user is on the dashboard
Actions : display user information, allow editing of profile picture, display name, password
post condition: updated user information
Acceptance tests: New user information is added to the database.
Iteration 2