Kyle Garber

CS 491

**CS 491 Engineering Notebook**.

**1/17:**

* Team Meeting:
  + Reviewed status of the project
  + Arranged sub teams
    - Carly and I are predominately on UI, with my focus being the FO interface

**1/19:**

* Team Meeting:
  + Reviewed upcoming vision statement and backlog assignments
  + Visualized project updates based on UI design proposals Jesse created
  + Begun work on backlog items in sub teams
* Begun work on combining the fo and tc login pages by moving them to the home Django app

**1/20:**

* Spent time considering current design and possible directions for the project since customer provided availability for meeting

**1/22:**

* Continued design contemplation as customer meeting had been scheduled for 1/26

**1/24:**

* Team Meeting:
  + Juliana shared new information from customer regarding project goals
  + Team discussion on design implementations to achieve new goals
* Continued working on backlog items for FO UI, Documentation and Planning, FD UI, and Communications epics
* Review of revised mission statement for the project that more accurately reflects current designs and goals

**1/25:**

* Prepared questions for customer meeting tomorrow, 1/26
* Begun work on merger of the fo and tc login pages

**1/26:**

* Team Meeting:
  + Met with customer
    - Discussed mission length
    - Discussed mission design
    - Discussed control design
    - Discussed team design
  + Digested customer design changes as a team

**1/27:**

* Finalized merger of fo and tc login pages
  + All signup and login is now handled by the home django app
  + fo and tc users are then directed to their app based on user type

**1/30:**

* Researched ways of asynchronously updating page content and database values
  + Ajax appears to be the most popular solution

**1/31:**

* Team Meeting:
  + Continued discussing customer changes
  + Debated value of team over single flight director administration of a SimCraft
  + Received official task for handling Ajax implementation
* Continued research into Ajax

**2/1:**

* Created new models in fo.models to test out Ajax database manipulation
* Created new urls to host and handle Ajax requests and tests

**2/2:**

* Team Meeting:
  + Shared progress on Ajax prototyping
  + Discussed individual subsystem designs
* Registered new models in fo.admin to monitor values on admin page

**2/3:**

* Imbedded Ajax script into foHome.html to handle a button press

**2/6:**

* Imbedded more Ajax functions into foSim.html:
  + Handle press of a submit button
  + Execute periodically to update display objects

**2/7:**

* Team meeting:
  + Further discussed Ajax implementation and project design

**2/8:**

* Created handlers in fo.views that manipulate database objects when called by Ajax scripts

**2/9:**

* Team meeting:
  + Discussed Ajax progress and committed to including Ajax functionality in upcoming demo
* Changed database fetching of objects from name handle to pk handle for all objects without mandatory unique names

**2/10:**

* Passed arguments from Ajax functions to functions in fo.views to determine new values for database objects being manipulated by the view function called
* Experimented with success functions to be executed by Ajax after fo.views function returns.

**2/11:**

* Successfully passed data from frontend foSim.html web-page to fo.views, manipulated database values, fetched new values, and returned new values to foSim.html to be displayed without web-page reload
* Functionality of previous demo using django forms has now been completely replaced and improved upon via the use of Ajax

**2/12:**

* Cleaned Ajax calls and foSim.html for maintainability
  + Relocated javascript to separate files referenced with script tags
* Committed and handled conflicts for Ajax functionality

**2/13:**

* Directly imported jquery.min.js to library to reduce reliance on third party hosted utilities
* Removed deprecated models and urls used for testing Ajax

**2/14:**

* Worked on presentation slides
* Prototyped generic script to be executed from the tc django apps views.py script
* Prototyped database manipulation via script external to a django app
* Team Meeting:
  + Presented for Inc 2. Sprint 1
    - Demonstrated current Ajax functionality
  + Discussed simulation script starting conditions

**2/15:**

* Prototyped new django app simapp to take over sim database functionality currently handled by tc
* Experimented with new input types on foSim.html

**2/16:**

* Team Meeting:
  + Discussed simulation script-starting
  + Discussed subsystem database design
  + Discussed fo input types to be hosted in foSim.html
* Added Ajax handling to new input types
* Finalized and committed new simapp django app
  + Changed references to tc.models sim functionality to simapp.models
  + Reset database and handled merge conflicts
* Renamed SWA/Simulation to SWA/simulation to follow naming conventions

**2/21:**

* Team Meeting:
  + Discussed Test Conductor UI design
  + Discussed database layout for missions in relation to sims and classes
  + Discussed various input types
  + Discussed standardized fo console layout
  + Discussed sim operators assignment

**2/23:**

* Team Meeting:
  + Discussed sim and mission creation UI
  + Discussed backend mission creation
  + Discussed backend sim initiation

**2/25:**

* Implemented page protection for fosim page in fo.views.fosim
  + fosim page is now only accessible by FlightOperator users that have the sim in their simlist.
* Fixed bug where error is thrown if user that tries to login does not exist

**2/26:**

* Implemented login\_required decorator on views that should be login protected in fo and tc Django apps
  + If a user is not logged in when trying to access protected pages, they will be redirected to the login page and sent to their desired page after successful login

**2/28:**

* Team Meeting:
  + Discussed mission script provided by customer

**3/1:**

* Begun work on implementing a command line interface for flight operators to use for SimCraft interaction

**3/2:**

* Completed and committed fo CLI
* Modified some models and ran migration tool
* Team Meeting:
  + Discussed implementation of subsystems with respect to new information
  + Discussed implementation of new user interface requirements

**3/4:**

* Begun work on some css issues present due to style sheet locality

**3/5:**

* Begun combining fo and tc functionalities into testapp to use in presentation demonstration
* Team Meeting:
  + Discussed benefit of having testapp updated for development purposes
  + Discussed mission and sim creation UIs
  + Discussed status of Simulation scripts and drivers

**3/6:**

* Fix bugs related to sim creation in testapp
* Tested application for use in demo presentation on 3/8

**3/7:**

* Presentation:
  + Worked on presentation areas for me to present
  + Practiced feature demo of new flight operator console with database integration
  + Presented with the team
* Helped team with some sim creation issues in tc app that I had fixed in testapp app
* Updated section 3 of the test plan

**3/21:**

* Team Meeting:
  + Discussed vision differences in SimObject operation
  + Considered running sim in volatile memory instead of representing full object in database
* Looked into methods of running sim without locking python execution

**3/22:**

* Found python thread module that may be able to execute sims without locking execution
* Experienced with thread creation in python sandbox

**3/23:**

* Team Meeting:
  + Debated pros and cons of running sims as standalone python objects
  + Discussed possibility of accessing sims under new architecture
* Attempted to create python threads within local Django server environment

**3/24:**

* Continued applying thread research to Django, namely creating threads from a Django view such as simapp/views.newSim()

**3/25:**

* Finished converting SimObject.py SimObject class to a python threading.thread object
* Implemented new function to check if sim is still active
* Looked into restarting sim if Django execution is interrupted and sim thread is killed

**3/26:**

* Extracted fo console html to standalone file foConsole.html
* Removed fo console html from subsystem html pages
* Committed all improvements made over previous weeks
* Merged conflicts and squashed bugs
* Protected tc/views functions with user type requirements

**3/27:**

* Worked on making sims accessible via their thread\_id stored in the Sim model as sim\_identifier
* Improved check function to return indication if sim is accessible when challing function on it from a Django view

**3/28:**

* Team Meeting:
  + Committed fully to implementing sims as threads
  + Since sims will be threads, new database infrastructure will need to be implemented to store user commands and sim responses
* Begun work on implementing string buffers for display and for each fo’s commands in Sim object model

**3/29:**

* Finished implementing display and command buffers in Sim object
* Add command string to an fo’s command buffer when entered in the fo’s console
* Make fo console specific to an fo, no matter which subsystem’s attributes they are viewing

**3/30:**

* Team Meeting:
  + Discussed new command buffer and related input functionality
    - Next step is inputting command options in fo command buffer
  + Discussed project bugs that had been found
* Fixed bug where test conductor objects had to be created for staff users manually in the Django admin page
  + Test conductor objects are now created automatically if none exists for an admin user when accessing the tc/home page
* Removed jquery.slim import in index.html that overwrote jquery import in foConsole.html

**3/31:**

* Started implementing sim thread access to display buffer by adding display string to buffer every 5 seconds while the sim thread is accessible

**4/2:**

* Generalized fo console input to work for each fo type
* Improved ajax communication to include a sim’s display buffer and command buffer for specific fo between fo/views.fetchdata and fosim.js

**4/3:**

* Changed get sim calls to get sim by pk instead of sim name across the project
* Fix bug in simthread.update() where it was not registered as a method

**4/4:**

* Team Meeting:
  + Discussed remaining tasks for the project
  + Discussed fosim console implementation changes
  + Discussed common html/css bugs
* Fixed css overlap bugs by importing file-specific css at lowest level instead of in index.html
* Changed subsystem html pages to include a scrollable div for attribute display
  + This fixes issue of not being able to see subsystem attributes based on console height

**4/5:**

* Begun work on pulling fo command options out of nested dictionaries in python subsystem objects

**4/6:**

* Team Meeting:
  + Discussed command list for each flight operator
  + Discussed simplifying method of storing a SimObjects attribute (removing dicts.py)
  + Discussed merging flight director and payload roles
* Implemented fo command lists