**Project Architecture**

Project Name: Graffiti Incident Tracking System

Team Members: Michael Jannain, Julian Nunez, Jimmy Doan, Jose Garcia

Date: 3/23/2016

Mike

Jose

Jimmy

Julian

1. Cover Page identifying the name of the report, project name, team members, and date of report.
2. Overview of project and understanding of requirements, including summary of requirements changes (if any)

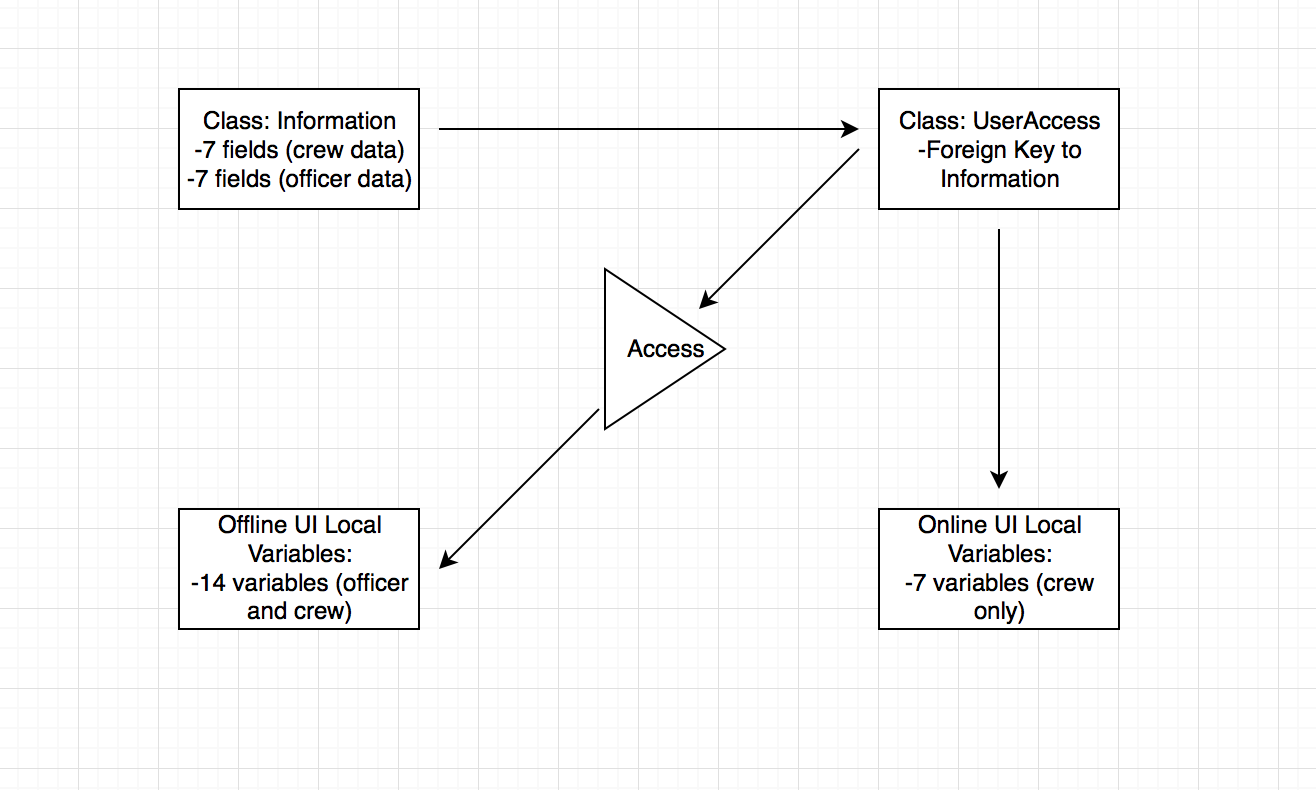
**Graffiti Incident Tracking System** - developed for Central City Public Works personnel and Law Enforcement Officials. Enables Public Works personnel to document graffiti info and send it to investigating Law Enforcement officials through an Internet Portal. Allows Law Enforcement officials to identify “hot spots” and view/edit/update graffiti & suspect info through a User Interface. Authorized personnel will also be able to request, view, or print reports based on information in the database. No changes to requirements.

1. Short discussion of any JAD sessions or prototypes developed to understand/evaluate requirements and decide on design.

1 - 2 hour long meetings (in person) every other week to review known GITS requirements, user expectations & user desires, divide tasks for upcoming deadlines/assignments, and discuss software plan & life cycle schedule. All team members are actively participating in discussions and have computers ready to quickly capture all information on a shared google document. Previous documents such as the Requirements Database & Analysis, Test Plan, and Project Notebook are used to make educated decisions on software design and planning.

1. Top Level Design

* Top level architecture diagram for subsystem, including rationale for architecture style selected—diagram must be specific to your project. This means that your components must have names meaningful to your application. Discussion should describe the purpose and scope of each component as well as the connectors between the components.
  + Model:
    - Class: Information
      * incidentName: records the name of the incident
      * Name: name of the crew member
      * Supervisor: name of supervisor
      * DateOnSite: date the cleanup occurred
      * ScaleOfCleanup: scale
      * DatePosted: date posted to site
      * CrewID: id of the crew that cleaned up
      * incidentID: incident id
    - Class: OfflineInfo:
      * OfficerID: officer’s id
      * Name: name of officer
      * Hotspot: is this location hot with crime
      * AnalysisPriv: sets scope for how much the officer can view and or edit from the analysis of data
    - Class: UserAccess (foreign key = Information)
      * UserAccessLevel: this is the foreign key related to the Information class - it decides if the user has access to the information and which of the variables.
  + Views: This describes the methodology of how the information in retrieved through GET/POST http requests from the database to the offline gui.
    - Lays out the format of the http address hierarchy for the website (e.g. /admin/ or /information/).
  + Web: Html formatting only (index.html)
* Database Design – Entity-Relationship Diagram (ERD) showing tables, attributes and relationships between tables. Also includes a database dictionary that defines all of the tables, fields/attributes, primary keys, foreign keys, indexes, and relationships.
  + Primary Key: incidentID (class Information)
  + Foreign Key: userAccess (class UserAccess)
  + Indices 1-7 onlin, 7-14 offline



* Requirements mapping to architecture components (this should be provided in your separate requirements matrix/repository).
  + Data Requirements → Information Class
  + User Access Requirements → UserAccess class
  + UI Requirements → Tkinter/html
  + Performance Requirements → Testing

5. Updates to Project Plan

* Estimates
  + Updated software size using the same methodology you used for the Requirements Analysis (if you change methodologies, then you need to convert your Requirements Analysis estimate to the current size units). Software size must be provided by “component”, depending on the methodology selected.
    - Web Portion Component: 750
    - Backend Python/Database Component: 750
    - Total: 1500
  + Testing software to be developed
    - Software size for any test tools to be developed
      * Identification of the sizing method selected:
        + SLOC method
      * Size of components (components defined by the sizing method selected… for lines of code, use top level design components)
        + 750 Lines of code for the top level (formula explained above)
      * Complexity factors associated with components
        + Top level = networking and integration of the web and offline database
* Schedule (OpenProj preferred, if available…. Otherwise, use spreadsheet)
  + Updated schedule showing planned start date for each task, duration of task, any predecessor tasks, and critical path
    - Schedule:

-Interactive Development (Start=2/1/16, End=4/4/16)

* + - * 1. Build 1 - Development (Start=2/1/16, End=3/10/16)

-Development (Start=2/1/16, End=3/10/16)

-Unit Test (Start=2/1/16, End=3/10/16)

-Handover to Test (Start=2/1/16, End=3/10/16)

* + - * 1. Build 2 - Development (Start=3/11/16, End=3/24/16)

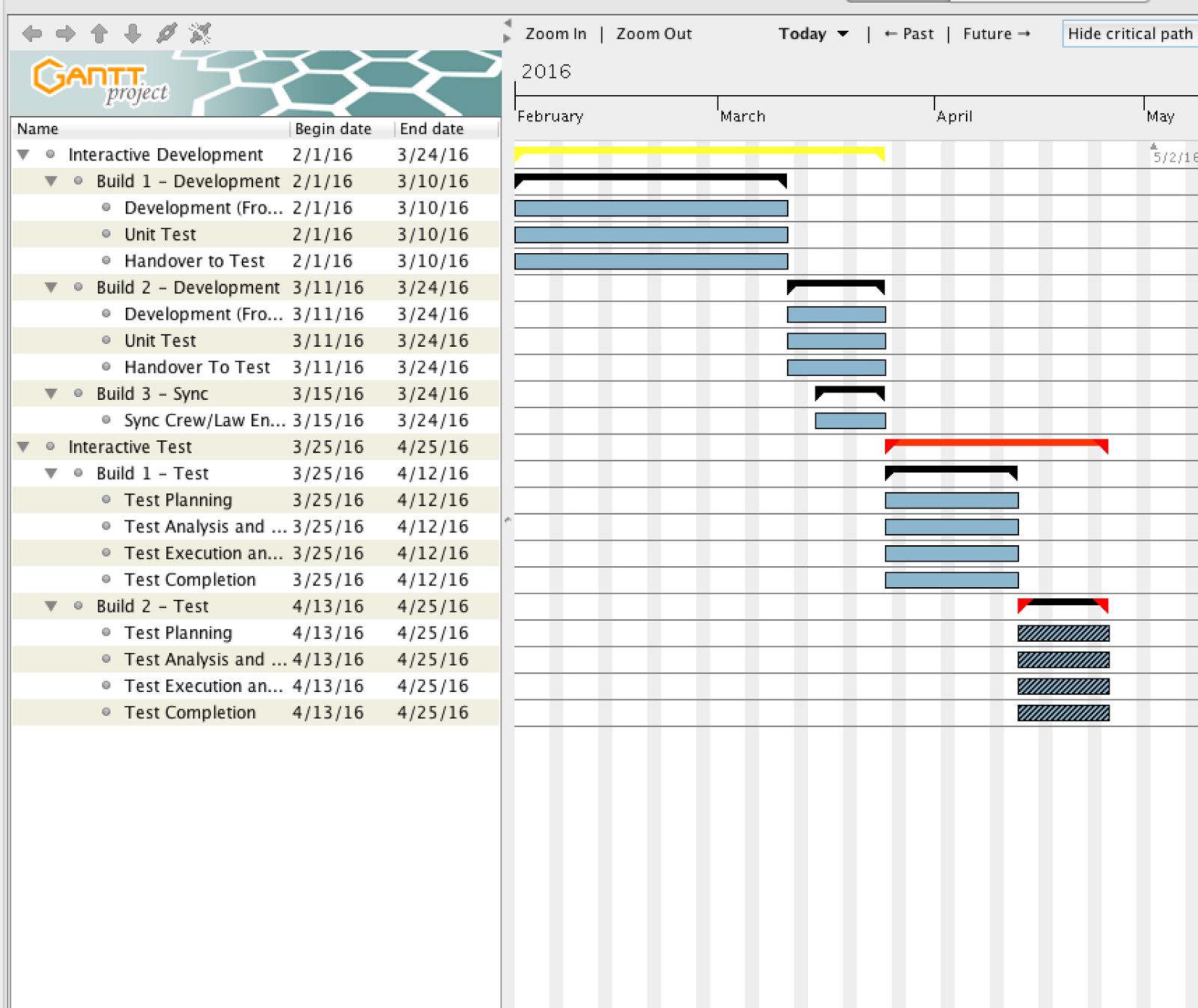
-Development (Start=3/11/16, End=3/24/16)

-Unit Test (Start=3/11/16, End=3/24/16)

-Handover to Test (Start=3/11/16, End=3/24/16)

* + - * 1. Build 3 - Sync (Start=4/4/16, End=4/24/16)

-Sync City Workers/Law Enf. (Start=4/4/16, End=4/24/16)



* + Progress to date with respect to the planned schedule
    - Finished Build 2.
    - Entering Build 3 - Sync (Start=4/4/16, End=4/24/16)
* Estimate of total number of labor hours to perform all tasks in the schedule
  + 4 people x 2 hrs per day x 53 = 424 hours
* Defects – number detected per phase (requirements, top level design, test planning)
  + Average defects: 0 defects - 2 phases so far → 0%

6. Identification and status of both previous and current/new issues/risks.

* Need to implement tracking system to GITS even though Build 1 & 2 are technically over. Risk is increased because this feature is implemented later in the life cycle than anticipated.
* Build 3 is pushed back and will take place later than anticipated so the risk has increased but there seems to be enough time to complete it without problems.
* Another risk has been using technologies that are new to us (programing languages), which has pushed our build 3 back.