Final Report Document

CS3200

Eli Abidor

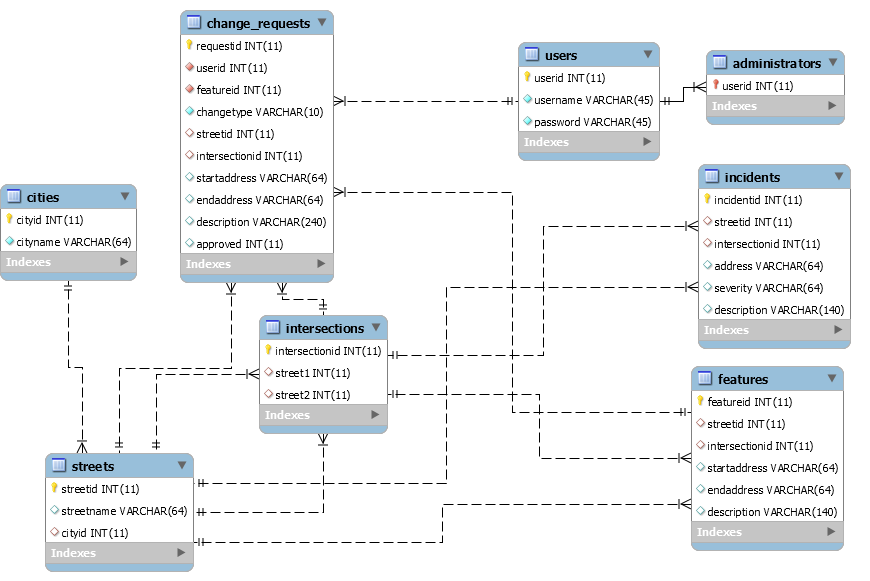
Gunnar Boehm

1. Readme:

Will need MySQL connector library, Python 3.5 interpreter

All files will be in same included folder

1. Flow and Schema



c

No

Change Request Denied

Yes

No

Admin

User

Access Denied

Log In

Validate

Administrator

User

No

Yes

Yes

Make Requested Database Changes

Approval

Request Feature Change

Return Incident Info

Closed

Incident Details?

Return bike feature data for given route

Check all streets/intersections for database entries

Request Route Cycle Info

1. Final User Flow:
   1. Enters username and Password
   2. If log in successful, prompted to “View” or “Change”, user inputs one of these options
      1. If user selects view, and user is an admin, can input “yes” to view change requests or “no” to view features
         1. If viewing change requests, admin will be asked to view a new request, inputting “new”, or a specific change request id, inputting that id
         2. Program returns change request details and admin approve by inputting “yes” or “no”.
         3. If viewing features, can input “street” or “intersection”
            1. Prompted to enter street name or 2 street names for intersection
            2. For either option, user can choose to look at “features” or “incidents”. Results for that street and feature type will be printed out, if any.
            3. After viewing, can choose to view again using “yes” or “no”
      2. If user selects change, can input “street” or “intersection”
         1. Prompted to enter street name or 2 street names for intersection
         2. Program prints out features for that street or intersection, with feature id’s listed
         3. Can choose to input “new”, “update” or “delete”
         4. If creating new feature in street, prompted sequentially for start address, end address, and description. Start and end address can be left blank
         5. If creating new feature for intersection, prompted for description.
         6. If updating feature in street, prompted sequentially for featureid (which can be looked for in printed list of features), start address, end address, and description. Start and end address can be left blank
         7. If updating feature in intersection, prompted sequentially for featureid (which can be looked for in printed list of features), and description.
         8. If deleting feature in street or intersection, prompted for featureid
      3. Once done, will be prompted if another view of change is wanted or to stop, with a “yes” or “no”. If no is selected, connection is closed and system exits. Otherwise repeats to step b.
2. Lessons Learned
   1. Team member Eli learned python from scratch
   2. Learned proper constraint usage and cardinality usage from trial and error
   3. Learned how to use efficient schema without excessive number of fields in a relation
3. Future Work
   1. Planning to use this database for an application for biking enthusiasts and commuters that can view and contribute information on city biking features
   2. Added functionality will include a GUI interface, either through an applet or a smartphone app or website. Additional functionality for reviewing the features, change requests, and other relations will be added. An ability to add or delete user accounts will be added. Will also possibly integrate google maps and biking directions API so routes can be grabbed between destinations from there. Will add ability to check features on whole routes, not just one street or intersection.