20th June 2017

1. Figure out:
    1. High Priority - Dead Code
        1. models.py:
            - HighestScore
            - LeaderItem
            - Leader
            - Dead code in `Bill` -- commented code
            - BillLineItem
            - TaskProgressUpdate
            - SiteConfiguration
        2. Review `views.py` and find out which functions are not being used
    2. Low Priority - Dead Code
        1. `Bill` model's `swachh_bharat_cess_pc` etc
        2. Check if `cumulative_logs` is being used?
    3. Dead Packages -- packages that we don't need anymore
    4. Learn more about existing DB models:
        1. Estimate
        2. EstimateTask
        3. EstimateTaskAssignment
2. Build an DB Diagram {Models and their interconnections}
    - First on Paper
    - If possible something like this
3. Figure out more on
    1. For `@cached_property` find out what is the validity and how it really works
    2. `SumWithDefault` what are aggregates and why are they used
4. Refactor:
    1. `calculate_actual_contribution` for model `Project`
5. Document all `classmethods` and `property`
6. Document logic of all big `if` conditions
7. Have different review session for `performance_report.py` and `views.py`

8th Dec 2017

1. Use of timezone.now() in place of `datetime.now()`
2. Create instances of `filtered` querires instead of repeating them again

Notes from Code Review Conducted at time of taking Project

1. Key Applications include the following:
    - e: This is where heart of the project lies. Most of the models and views reside in this project
2. Business logic is currently split in couple of places.
    - Helper Functions within Model Classes
    - Some inline Javascript withing HTML files
3. All setting related to project are stored in `settings.py`. Some of the setting are derieve from environmental variables
4. JSON Web Tokens are used for authentication
5. Few 3rd party application that are used are as follows:
    - Crispy Forms
    - wkhtmltopdf is used for generating PDFs
    - django-nvd3 is used for charts
    - Django Rest Framework is used for APIs
6. The code is written using Function Based Views
7. JsonResponse is used in views probably for Ajax like functionalities
8. Static assets are stored in `static` directory
9. All templates are stored within `e/templates`, with reporting related templates getting stored in `e/templates/reports`
10. Most templates extend `base.html` file
11. There are few ad-hoc scripts that are stored in `scripts`
12. Bower is used for package management {needs more investigation}
13. There are few templatetags that are used e.g. `active` etc which are located in `e_extras.py` file

Key Entities

1. UserProfile: Record here is created for each user on our application. This is where few of the following fields will be needed to be added:
    - PAN/TAN/GST
    - expires_on - This will be required for our SAAS offering. We would need to probably associate some sort of Foreign Key for grouping accounts SAASAccount.
2. Department: This can be thought of Line of Business, Profit Center etc. Right now a Department has a single Head and single Admin. This is where change needs to be peformed for allowing multiple heads
3. Customer: Client for whom the billing needs to be done
4. Job: A job by definition is collection of related Tasks {M2M Relation} and a single Department
5. Task: This is unit of work on which a Log can be created {Entry in some sort of Worklog}
6. Estimate: An estimate is created for Customer for a specific Job and can have a Leader who can be different from Deparment Head or Admin.
7. Estimate Task: Is a unit of work that can be attached to Estimate and Task. manner field is also included in this model. Upon an Estimate getting approved all enties from Estimate Task get copied over to Project Task
8. Estimate Task Assigment: These set of records capture to which EstimatTask are assigned to which User. Similar to Estimate Task, all entries from Estimate Task Assignment gets converted to Project Task Assignment
9. Project: An Estimate gets converted to Project once its approved. All the metrics like cumulative_days, predicted_days etc are stored in this model.
10. Bill: This is akin to Invoice, all billing related information is stored in this model.
11. Bill Line Item: Aggregate information are stored in Bill model. BillLineItem are itemized details for a Bill.
12. Log: This is an entry that captures a unit of work done ProjectTask. Few things note, a Log can be categories into one of the following
    - Office
    - Idle
    - Leave
    - Sunday
    - Bank Holiday/Festival
13. Task Progress Update: This is where progress on ProjectTask is stored
14. Site Configuration: This is where billing related information is store. TBD: Need to investigate where exactly is it used
15. Trophy: These are badges that a user get to see on their profiles. Think gamification, think Trophy

Road Ahead

1. Check feasiblity of this solution to retain values in all forms after validation errors
2. create_log is where logic for Trophy can be changed. Suggested place to have all business logic in models.py file
3. Investigate what JS plugin was used for Drag and Drop, check what missing
4. See if we can integrate Angular or Vue.js for tasks like allowing users to add more task within Job Templates
5. Do UI audit and make sure we fix responsive issues. Bootstrap has been used, but some minor changes need to be ensured
6. Get rid of print statements, see if we can integrate caching using redis
7. Few UI and logic changes need to be performed on Department Details page
8. Bulk Import section needs to be created, which will allow loading ODOO like data for different models. This would include:
    - Customer
    - Department
    - Employee
    - Job Template
    - Log
9. In existing Reports we need to provide Export functionality. Tablib can be looked up for this functionality
10. Few items need to be added in settings.py file, which would have an impact on Report. Fields would include:
    - min_log_time_hrs
    - min_log_time_min which will allow us to - classify employers are defaulter or not {Default Value: 8}
    - max_allowed_users
11. To allow multiple department in model Department we need to make head as ManyToMany field
12. To keep track of rate changes for an employee create RateChange model with following field:
    - user: ForeignKey {User}
    - changed_on: DateTimeField
    - initial_costing_rate: IntegerField
    - updated_costing_rate: IntegerField
    - initial_billing_rate: IntegerField
    - updated_billing_rate: IntegerField
13. Locking needs to implemented

    - In a given application has created more than `max_allowed_users`
    - If current date is greater than one specified `SAASAccount`
    - Other locking rules TBD
14. `Delete Project` option that has to be implemented, will need `archived : BooleanField`. This will mark Project as deleted but will never delete the project. I will hide them from list of active projects
15. Check if django-helpdesk can be used as a variant for ZenDesk
16. Check if django-activity-stream can be used for keeping tack of changes to Department Heads/Project Leads only
17. Implement `Forget Password` feature