horizontal line

Product Requirements Document

Date: 19th April 2020

Document Version: 1.0

Version Description: First Draft

Updated By: Ranon Sim

Reviewed By:

Change Description:

# OVERVIEW

This document describes the development of a Fraud Detection System for ABC in order to reduce the number of fraud cases happening to drivers and riders. The following will be covered:

1. Describe why fraud is a problem and the objective/goals of this project
2. Explain why Machine Learning is helpful for fraud detection
3. Describe the development working fraud detection ML model for the regional ops management team to monitor and manage fraudulent activity caused by fake accounts
4. Go through the model development plan, data features used and approach to evaluate model performance
5. Discuss the tasks involved as well as the short-term and long-term roadmap for the project
6. Discuss the business resources required

# BUSINESS PROBLEM - FRAUD IN ABC COMPANY

With millions of transactions being processed for over a million monthly users, the number of moving parts in our platform presents opportunities for fraudsters to abuse and undermine our system for their own benefit. Fraudsters can range from individuals users to organized crime syndicates who use advanced technology to circumvent the system and reap rewards. Frauds that occur on our platform include payment fraud, incentive abuse, social engineering and account takeovers. These frauds not only have adverse impacts on our financials but also hurt user experiences.

# 

# OBJECTIVE

Come up with a plan to detect fraudulent behavior in order to reduce the number of fraud cases on our platform

# GOALS

* Design a framework for fraud detection that includes a machine-learning (ML) based solution
* Build a working fraud detection ML model that can deliver value for the regional operations team

# STAKEHOLDERS

|  |  |
| --- | --- |
| End-user | Regional Operations Team |
| Data Scientists | Timmy, Joel |
| PM | Ranon |

Other Stakeholders that may be required

|  |
| --- |
| Analysts |
| Data Engineers |
| System Engineers |

# SCOPE

Whilst there are many types of frauds occurring, this project will focus on **identifying fake accounts** created by fraudsters for various illicit activities e.g. offering ride services in the black market or abusing the performance incentive systems.

# PRODUCT DESIGN OVERVIEW

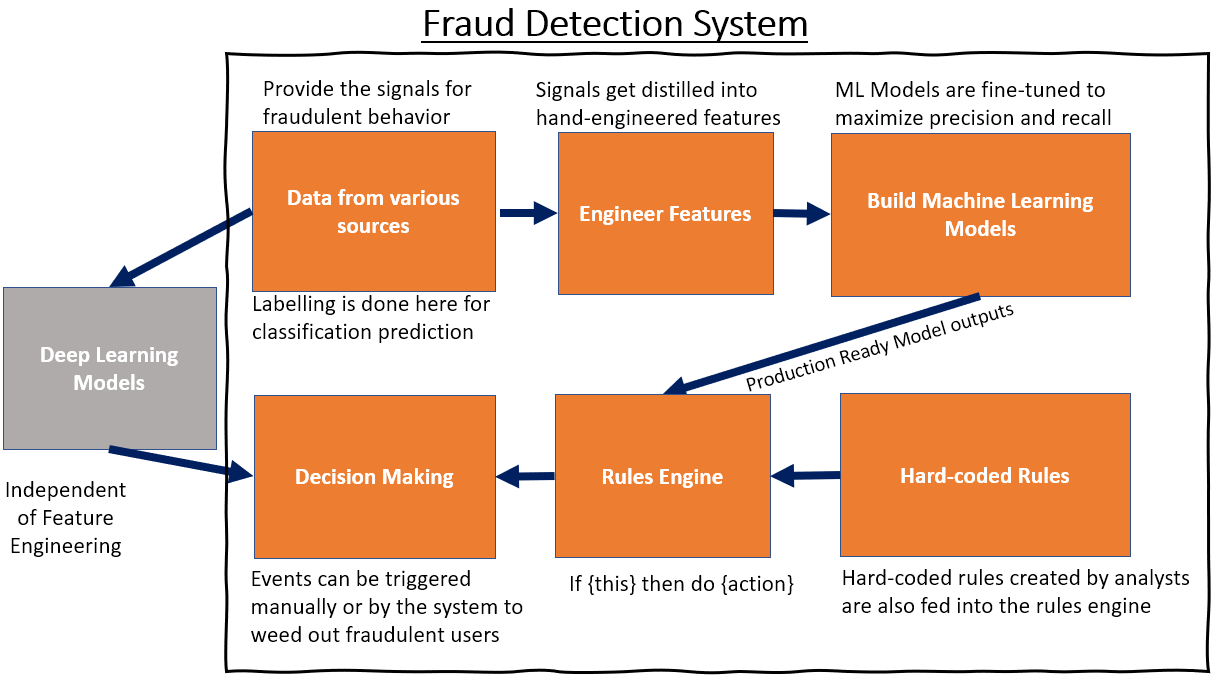
## Fraud-Management

Before we jump into details on fraud detection, we should realize that fraud detection is one component of how fraud is being managed. Overall, the components include fraud detection, fraud analysis and decision support.

Fraud-detection itself can have many applications across the various business functions of the company, and be solved using different old school (rule-based) and modern (machine learning) techniques.

## Fraud-Detection Workflow

Our approach towards fraud detection is a mix between handcrafted business rules defined by the business analysts and machine learning models developed by our team of data scientists. These models make decisions that trigger events (e.g. pre-authorizations and identity challenges) that aim to weed out fraudulent users.



# 

# MACHINE LEARNING MODEL

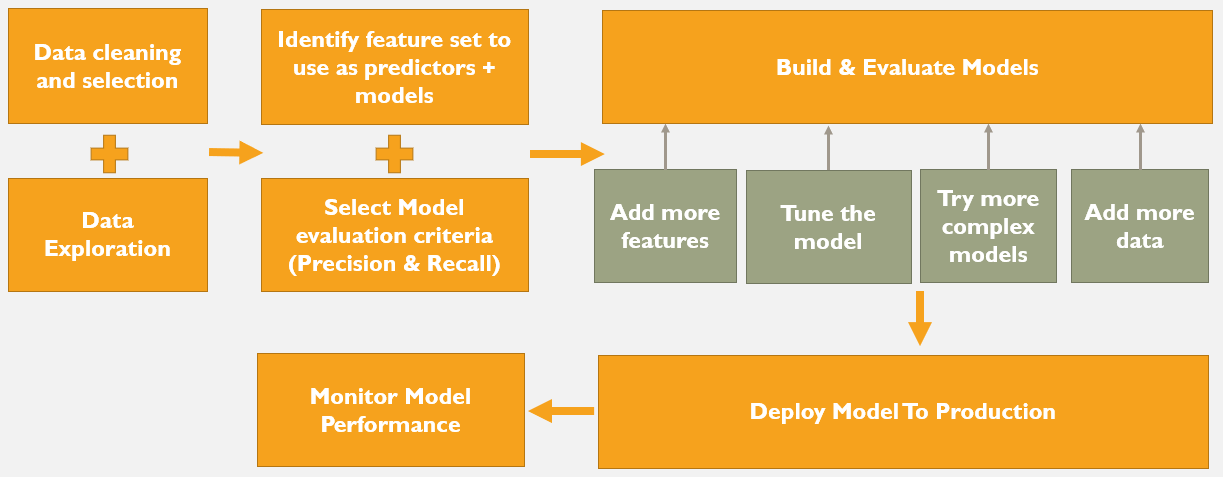
## Objective of Machine Learning (ML) Model

As rule-based systems become increasingly complex, we turn to ML for a more robust, scalable solution. The ML model will make use of features generated from signals indicating odd behavior to identify fake accounts created by fraudsters.

## Machine Learning Model Pipeline

One of the critical steps in developing an ML Model is to ensure a proper pipeline is established so that the model can be retrained reliably and consistently over time as better features and new data gets added to it. A typical ML pipeline looks like this:

Basic Workflow:



## Features used in Machine Learning Model

**Intuition when generating features:** Fraudsters are going to behave differently when they interact with the mobile app compared to normal users because their agenda is different.

Going by this intuition, we can collect ***user event data at different stages of the user’s life cycle*** to create features. The following are some examples of user event data (not exhaustive):

* **Signup information as features:** During account creation, we can already collect 4 features at signup and 3 more when the user adds a card and requests for a trip. The figure below describes signals that are often combined with business logic to create rules that detect fraud. For example, a user whose request location IP and sign up IP is completely different is unusual and may be flagged out as fraudulent.



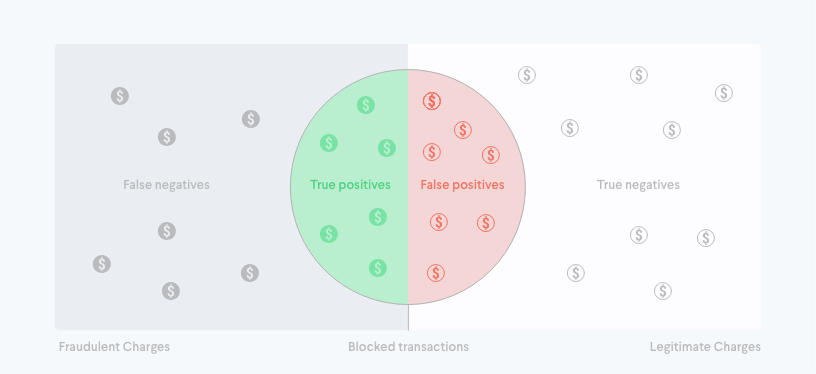
* **Behavior Sequences as a feature:** Another way to identify fake users is by looking at their sequence of behaviors. For example, the recurring pattern of cancellations (calculate the average cancellation rate per user) from a user after the requested rides are dispatched in a short amount of time might be an indicator of fraud. Other possible behavioral sequences to analyze include non-completion rates of rides, contact rate between a user and a driver and recurrence of edits of the drop-off and pick-up locations.
* **Time-Series features:** Having a time-based rolling window of ride occurrences and costs can also be useful to monitor if fake users start spending a lot more than usual after a period of time.

## Evaluating of model performance

### **Defining the Model Evaluation Metrics:**

****

* True positives (how many fraudsters we block)
* False positives (how many good people we block)
* False negatives (how many fraudsters we allow)



A model is deemed to perform well if it is able to identify as many positive cases correctly. Ideally we want to maximize positive detection rates but at the same time not tag legitimate users as fraudsters. This requires a balance between high recall and high precision. The choice to prioritize one over another requires weighing the cost of false positives to false negatives for our company.

# SHORT-TERM ROADMAP

## Goal

***Build an MVP to tackle fake accounts that can be deployed and evaluated in real-time by the regional operations team.***

## Prioritization of Tasks

|  |  |
| --- | --- |
| **Step** | **Task Description** |
| **1** | Set a workable timeline to deliver the MVP and plan for release and deployment date |
| **2** | Set up a meeting between relevant stakeholders to identify a diversified set of data sources that can be used for analysis. |
| **3** | Design a data collection framework to gather the data efficiently and ensure the data used to make fraud reduction decisions will flow accurately and consistently |
| **4** | Perform data preparation, including the labelling of data (if not yet done) as fraudulent/genuine for the classification model |
| **5** | Conduct data exploration phase with the purpose of   * + Understanding user behavior   + Identifying whether existing business rules/policies can be / are being exploited   + Identify features capturing behavioral patterns signalling fraud   + Further identify behavioral patterns that aren’t easy to hide   An example of a machine learning approach to understand user behavior is called clustering which can be used to discover organized fraud rings or unique behavioral patterns. |
| **6** | Build version one of the ML prediction model using the engineered features |
| **7** | Test and evaluate the model against a test dataset (dataset that the model has not seen or trained on). This will be used to assess whether the model passes our success metrics |
| **8** | Work with the regional operations department and system engineers to deploy model into production according to existing infrastructure needs |
| **9** | Continue to monitor the model’s performance every month. If the performance fails below the SLA defined by the end-user, the data scientists will   * Refresh the model using new data and evaluate its performance * Brainstorm new features / rules that can be incorporated in the model * Build a new model |

# 

# 

## Short-Term Success Metrics

* **Functionality** :
  1. Output of Model must be able to explain why an account is fake and facilitate decision making
* **Performance**:
  1. version 1.0 of model must be able to achieve at least 0.7 recall (TP / (TP + FN) ) or identify 70% of the total number of fraudulent cases (from the validation dataset)
  2. Application must be able to generate results within X minutes / Y hours (to be discussed with the regional operations department)
* **Usability**:
  1. Model should not be a black-box i.e. the reasons for a fraud occurring should be interpretable.
  2. False positive cannot exceed 50% (measured using precision ) and impact non-fraudulent users’ experience
  3. Data scientists should be able to provide the necessary information for the operations team to identify, monitor and take action on scammers/suspects.

## 

## Development Stages

### **Stage 1 - Prototype a simple, interpretable classification model + Focus on building good baseline features through feature engineering**

Stage 1 development is meant to create a baseline model that is highly interpretable whilst focusing on creating the most useful features that can distinguish between a good and fraudulent user. Here’s what to expect at this stage:

|  |  |
| --- | --- |
| **Model(s) Tested** | Logistic Regression (Baseline) |
| **Deliverables** | * Creation of Features from various data sources * Relatively simple code base from which new features can be prototyped and tested * Feature importance for each prediction |

### 

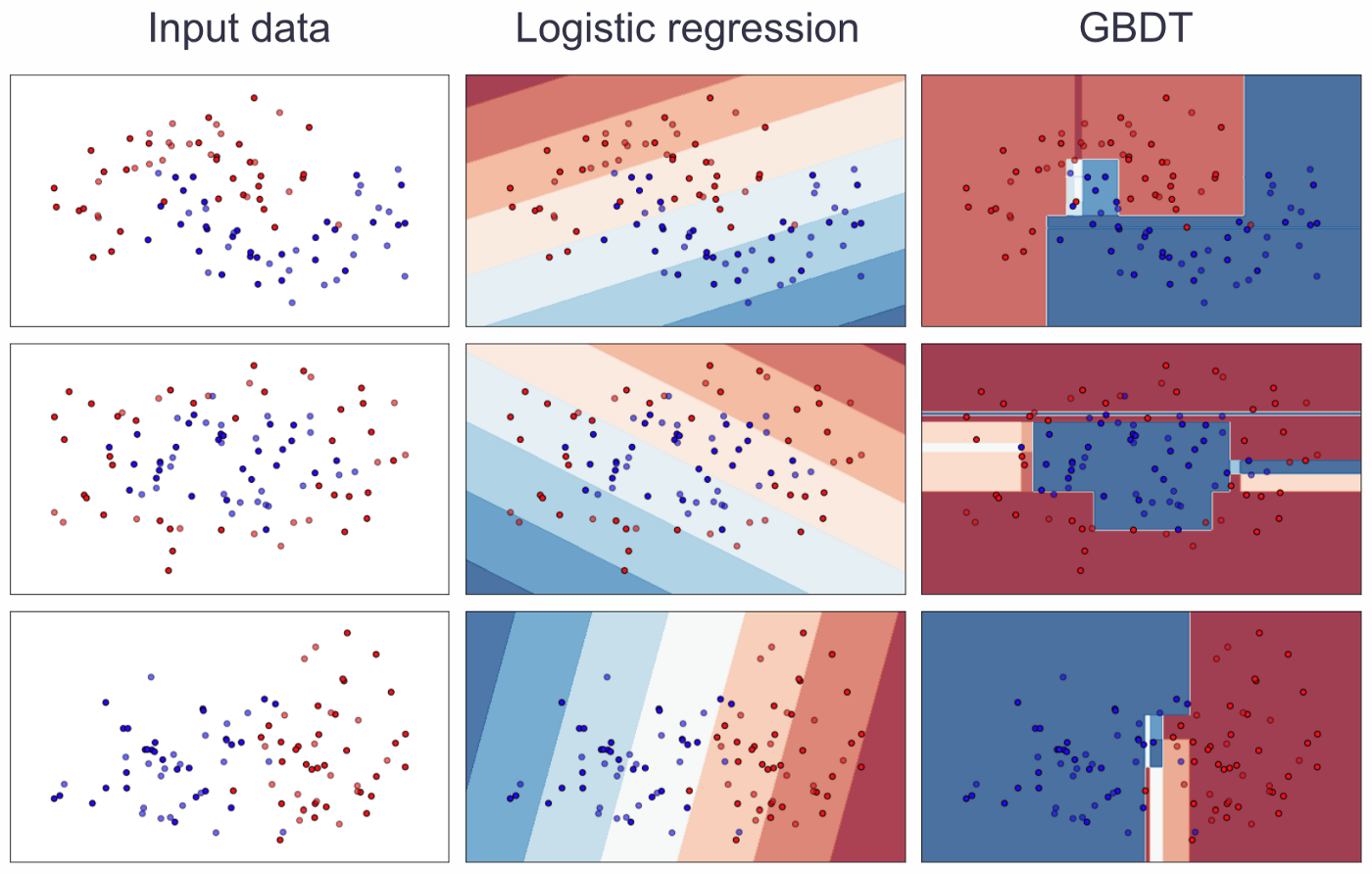
### 

### **Stage 2 - Build a more sophisticated model that can handle more complex features**

As fraudsters become more sophisticated in their behavior to avoid detection, it becomes harder to engineer features for the Logistic Regression model (that requires good features to have strong linear relationships with the likelihood of fraud).

As such, we will explore the use of more sophisticated models like **tree-based ensembles** that can better capture nonlinear relationships between features fraud likelihood.

To provide a simple explanation on the difference between Logistic Regression and a Gradient-Boosted Decision Tree **(GBDT)**, we can look at the following illustration:



The logistic regression decision boundary is simply a hyperplane on the feature space, which means good features must exhibit strong linear relationships with the likelihood of fraud. For GBDT, the decision boundaries are smoothed versions of a collection of higher-dimensional boxes, which allow us to encode more complicated feature interactions.

Here’s what to expect at this stage:

|  |  |
| --- | --- |
| **Model(s) Tested** | Tree-based and ensemble Models that have proven performance across multiple types of data |
| **Deliverables** | * Complex Hand-Engineered Features fed to new GBDT model * Working GBDT model that can output interpretable results |

### **Stage 3 - Productionizing the model**

The next step is to productionize the model and either output the results or feed the results of the model into a rule-engine for decision making.

The pathway towards production requires a **few considerations** put in place. Here are some examples:

* **Model outputs should be helpful and interpretable to end-users:** At the end of the day, even the most bleeding-edge model becomes useless when end-users cannot use or interpret its output for decision making. To preclude this from happening, we will

1. **Determine a relevant set of metrics** that the model will output before-hand
2. **Employ various** **model interpretability techniques**, such as *feature importances* and *model explainers* (e.g. shap values). Simply put, feature importances help us to prioritize which feature to put into the production model. Shap Values on the other hand help us answer the question: ‘why was the model wrong for this particular prediction?’ and ‘Which features influenced and resulted in the wrong prediction?’

* **Running predictions / generating output should be simple and repeatable:** Once a model is built, we need to ensure the model pipeline and model itself can be retrieved quickly and reused wherever possible. To do so, we will use code libraries that can allow us to easily import, export and run necessary items. We should also build certain functions to simplify the process of running predictions and generating output so that it can be reused later for different fraud problems. An example of such a code library is Scikit-Learn. However, there are new automated Machine Learning packages that do this for us easily too that are worth trying.
* **Make troubleshooting easier:** As development progresses, multiple versions of code packages models and/or model building tasks will appear throughout the development lifecycle. Proper versioning of these items is important to avoid frustrating errors arising from code inconsistency between development and production environments. We will thus set in place best practices in package, model and code versioning to adhere to.
* **Provide security:** Sophisticated fraudsters armed with enough knowledge can attack the very system that is meant to detect and prevent fraud given the chance. Meetings need to be set up with relevant teams (e.g. data scientists/security/dev-ops) to identify vulnerabilities in the production environment that could be exploited so we can prevent it from happening.

# 

# LONG-TERM ROADMAP (to be discussed in detail)

## Goals

* *To handle different types of fraud use cases and tackle engineering challenges*
* *Focus on improving the machine learning model and ensuring the model can scale up to support increasing amounts of data and feature complexity.*
* *Construct scalable solutions that make it difficult to commit fraud from the start: fraud prevention rather than just detection and mitigation.*
* *Reduce fraud-decision making time to a matter seconds*

## 

## Prioritization of Tasks

## 

## 

## 

## Long-Term Success Metrics (to be discussed)

* Functionality :
  1. Models are able to handle much more complicated features
  2. Models rely less on hard-coded features but can identify new user behavior on their own
  3. Models can learn good user behavior to detect when someone is deviating from it
* Performance:
  1. Application have high precision and recall (above 0.9)
  2. Decision making time must be reduced to a matter of seconds or sub-seconds
* Usability:
  1. Monitoring tool (dashboard form) should contain all the necessary information for the operations team to identify, monitor and take action on scammers/suspects.

# CRITICAL COMPONENTS FOR SMOOTH DELIVERY TO PRODUCTION

# Data: As with all ML applications, quality data is foundational to building anti-fraud ML systems. Data sets are only growing larger, and as the volumes increase, so does the challenge of detecting fraud. The make-or-break factor is having a ML platform that can scale as data and complexity increase.

# Multiplicity: There’s no single ML algorithm or method that works best for fraud detection. Success comes from the ability to try lots of different methods, testing variations and evaluating them with an array of data sets. That requires a toolkit with a variety of supervised and unsupervised methods, as well as a range of feature engineering techniques.

# Feature Pipeline: As the number and complexity of features increase at each stage, we will need to build a reusable feature engineering pipeline and add the new features to this pipeline so that the process can be automated and run consistently, regardless whether the model is offline or online (deployed)

# Integration: Only 50% of all models developed ever make it into production, resulting in a lot of wasted effort. Once you have developed a ML model, the challenge becomes deploying it in an operational run-time environment. If your data is in Hadoop, it makes sense that your ML model can be applied in Hadoop. Similarly, if your data is streaming in real-time systems, you want a ML engine that can run in real time or in stream. Portability of the model and integration of the decision logic within operational systems is paramount to stopping fraud at scale – and as it occurs at scale.

# Model Interpretability: Explaining the “what” and the “how” for ML systems is critical to help users identify and take action on fraudsters. This explainability factor is often referred to as “white-boxing” or interpretability, and it is critical for supporting model validation and governance processes.

1. **Ongoing monitoring:** As populations and the underlying data shift, system inputs are expected to degrade and affect overall performance. Newer ML methods can adapt to new and unidentified patterns as underlying changes occur. This eliminates some, but not all, of the ML retraining and evaluation steps. A good monitoring program registers and tracks the ongoing efficacy of all models.

# Experimentation: Successful ML programs have an element of ongoing experimentation. It isn’t enough to just build a ML model and let it crunch. Fraudsters are clever, and technology changes quickly. Having a sandbox where data scientists can freely experiment with a variety of methods, data and techniques to combat fraud has become a critical aspect of top anti-fraud programs.

# BUSINESS RESOURCE REQUIREMENTS

### **Resources required (not an exhaustive list)**:

People from the business, finance, economics, risk management and operations teams

### **Objective:**

To assist our data scientists, the services of the business units can be employed in the following ways:

* Provide knowledge on how fraudulent activities have occurred/are occurring.
  + The operations team can share what they learnt from the manual study of fraudulent user accounts e.g. from user reviews and ratings / behavioral patterns
  + People with finance and economic skills can apply their skill sets to identify trends and understand why it makes economic sense for fraudsters to operate
  + People can be employed to study the fraud black market such as how fraud rings operate in order to help us adapt our fraud metrics over time
* Help identify unique business functionalities and policies in our ride-hailing application that leave us susceptible to exploitation.
  + What Is the driver signup process like? Is the information required for onboarding a driver robust enough to help us identify fraudulent behavior?
  + Are logins currently done using face detection? If so, can the face-recognition software be tricked using a fake ID or photo?
* Provide a diverse set of data sources from which behavioral patterns can be captured.

# ASSUMPTIONS

* **Data:** The richness of the data; How we expect the data to be formatted, what features are available to us, how global in scope the data is, the applicability of a certain feature to act as a proxy for something, etc.
* **Models**: All Machine Learning models come with their own assumptions. For example, an assumption that our model can force on us is no NULL values, which may require us to include an imputation step in our tasks plan. These assumptions and any violation of the assumptions are important to log to make sure we fully understand the pros and cons of using any mode.
* **Workflow or Data Pipeline:** For example, we might assume that geographical location is available only as a categorical city/state, but our data pipeline might change to pick up latitude/longitude data in the future—and cause us to adjust our model in production.

# RISKS

* Underestimating the potential work
* Inaccurate identification of the task priorities and project benefits
* Unable to get hold of relevant stakeholders to discuss important components of the project

# OTHER DOCUMENTATION COMPONENTS (Not Included in this document)

* The Architecture of the fraud-detection system e.g. the backend data storage / development environments / modelling software / front-end API) etc. will be covered in further detail within the technical specification document.
* Timeline for all tasks will be in a separate Project Specifications Document