Software Development Plan

# Introduction

## Our goal is to provide high quality software for TES. High quality software is well-designed, maintainable, tested as fully as possible prior to release, and meets the company’s short and long-term needs and business opportunities.

## This plan is an attempt to understand as fully as possible both the short- and long-term goals of the company. Understanding these goals enables us to make good design and implementation decisions and allows us to focus our efforts efficiently. It’s also imperative to have company management and software development in sync and sharing the same goals.

# Current Applications

## Legacy TES Applications

### Internal applications built with direct coupling to SQL server, and only accessible from local network. Built using C++/Embarcadero, which is not a modern skillset that you can’t readily find additional developers to contribute. Tight database coupling is problematic for maintenance of both the application and the underlying database architecture.

### Mounts

### Poles

### Towers

### Inspections

### EPM

### Billing

### Maintenance

### Dashboard (mobile)

## Web Portal

### Built on ASP.Net which is a fading technology and is difficult to find development resources for. A lot of dependence on DevExpress controls for functionality which locks us into the advantages and limitations of their design patterns and capabilities. ASP.Net requires too much functionality embedded in difficult to maintain HTML/JavaScript code. Limited support lifecycle from MS.

## Mobile Contractor App (Android, iOS)

### Built using standard MS development tools on the .NET application stack. Deployable to standard Android and iOS client devices. Back end services built upon standard .NET application stack, Including IIS and SQL Server.

## SMART Tool

### Built using standard MS development tools on the .NET application stack. Deployable to standard Windows 10 client devices. Back end services built upon standard .NET application stack, Including IIS and SQL Server.

## Web API

## Admin Web App

## PMI Web Portal

## Ground Based Assessment (GBS) Mobile App

# Technology Platforms Description

## Our recent development provides a multi-tier architecture with separation of database from the UI, and secure Web-API access to data and services. This provides the opportunity to further develop Web, mobile, and desktop applications without limitations on how we deploy and scale the backend services.

## AWS

## IIS

## SQL Server

## .NET

## Windows Server

## Windows Desktop

## Android

## iOS

## Software Libraries

### DevExpress (ASP.Net, C++/Windows)

### GrapeCity Component One (UWP)

### GrapeCity FlexGrid (UWP)

### GrapeCity PDF conversion libraries (.NET)

## External Application Dependencies

### RISA

### MS Word

### MS Excel

# Infrastructure Description

## Self-Hosted Servers

## AWS

## Internal Network

## ISP

# Future Applications

## Additional Mobile Apps?

## New Maintenance App

## New EPM Tool

## New Poles App

## New Towers App

## New Inspections App

## New Billing App

# Database Architecture

# Future Goals

## Risk Mitigation

## Reuse and Maintainability

## Testing

## Sustainable Development Process

# Software Development Process

## Requirements Analysis

## Design

## Prototyping

## Test Process

## Delivery Process

## Iterative Development

# Additional Resourced Needed

## Full-time developer to assist with test, Web portal, mobile apps

## Contract help for GBA

## Contract help to start Web portal development

# Opportunity for Commercial Applications

# Current Software Development Priorities

## SMART Tool

### Fix bug with creating new Sector models in MA

### Add RISA model images to Open Mount Models page

### Accommodate new Mount Mapping format

### Cost Estimate spreadsheet

### Wind Speed Force Calcs

### Auto-unwiring of event handlers

### Replace Documents page thumbnail/grid bindings

### .NET Core 3.0

## GBA

### Add multiple photos per assessment step

### Improve styling of overall app

## TES API

### Database refactoring

## Verizon API

## Test Framework

### Web API

### Verizon API

### SMART Tool regression testing

## Web Portal

### Rebuild as Razor/Blazor pages on top of our APIs

## PMI Web Portal

## Admin Web Portal

### Release Verizon user management page to James

## Database

### Synchronization strategy between instances

**Testing**

* Automated testing
* User level testing
* Test process
* Resource requirements?

**Release**

* Zero downtime goal?
* Release process

**Redundancy Considerations**

* How to provide an architecture that’s fault tolerant?
* ISP
* Power

**Business Process Optimization**

* Remain reactive to business needs
* Additional automation
* Mount characterization
* Automated scheduling
* Additional reporting and data analysis
* Engineering analysis opportunities (guyed tower cable strength calculation, etc.)
* Additional efficiency opportunities?
* Maintain a competitive edge

**Development Timeline**

* Maintain balance between features, timelines, and resources

**SMART Tool Development**

* Sync AWS schema to TES internal schema mods
* Refactoring data models
* Formalize the release process
* Formalize testing process (with Maser and internal)
* 30 degree implementation
* Thumbnails
* User job locking
* Regression testing Web API
* Regression test internal classes
* Code refactoring
* Build machine
* Pre-release test environment (database)
* Other features/bugs list

**Development Goals**

* Remove the fourple concept from the data model
* Remove need to copy attributes between tables
* Remove data (history) loss by insert-only model; don’t use UPDATEs
* Provide referential integrity