Capstone: Task Pipeline

Western Governors University

000929868

Table of Contents

Prompt A

Letter of Transmittal

April 21th, 2020

Ms. JoAnn Miller

Seamus Company

650 N South Street

Dellberg, WI 99999

Ms. Miller,

Optimizing assignment of tasks is an obstacle faced by every service organization. This task is complicated by the distributed nature of institutional knowledge and the difficulty inherent in transforming that knowledge into actionable wisdom. Solving this challenge creates opportunities for improved operational efficiency as optimized task assignment reduces the time taken to resolve each task and reduces the risk of defects that require rework.

The task pipeline will provide a ranked list of resources whose experience is an optimized match for each task provided. The browser-based interface will allow exploration of the driving factors behind each ranking. Factors which will be extracted by processing the provided task descriptions into descriptive tags and incorporating existing descriptive tags about the

customer and the site. Due to the sensitive nature of the data processed by this pipeline, a relational database management system will be used for secure data storage and access. To preserve confidentiality of existing data sources, the demonstration dataset for the pipeline is randomly generated from generic sources then hand curated by an industry expert.

Providing task assignment recommendations based on resource proficiencies and availability allows a variety of practical efficiency gains. Service resources will benefit with better utilization of training and more applicable training. Dispatch staff will be able to more quickly evaluate tasks and select appropriate resources to resolve requests. Customers experience more consistent interactions with less disruption, fewer repeat visits and less time required to resolve tasks.

The resulting product will encapsulate a flexible and customizable browser-based interface, a demonstration database and a modular processing pipeline. The pipeline will accept a task description along with associated site information. It will combine institutional knowledge about customer systems, site-specific details and company resources to provide a ranked list of the employees able to most efficiently resolve the request.

Building, installing and maintaining an application of this critical nature will require an investment of $27,000 with yearly maintenance costs totalling to $5,400 or 20%. This pricing relies on the utilization of development tools and libraries available at no additional cost and with no associated licensing fees.

To implement this solution, I will be relying on academic training that culminated in a Bachelors of Computer Science, with proficiency validated through independent certifications from CompTIA, EdX, ITIL, Microsoft and Udacity. My academic training has been refined by more

than 10 years of experience in physical security service and a series of successful cross-vendor system integration projects.

Sincerely,

Project Recommendation

Problem Summary

Uneven distribution of institutional knowledge creates difficulties for dispatch staff as they perform their duties, assessing tasks and attempting to determine optimal task assignments. To control operational and warranty costs, adequate task assignment optimization must occur. Current methods of resolving this problem are numerous, each accepting reduced effectiveness to produce consistent results; training dispatch staff in technology and techniques that they will not otherwise need or use; limiting product variety and by extension, customer base; dividing technical resources along product lines; and enforcing standardized training without regard for individual resource talents or interests. As a result, only those customers, sites and situations where a costly incident has already occurred are likely to receive a higher level of analysis of need and available resources.

In resolving this, the solution will provide data input and review interfaces and data storage. It will require an initial labor investment for data input and maintenance as new relevant information becomes available or existing information becomes outdated.

Application Benefits

The task pipeline will provide a ranked list of resources whose experience is an optimized match for each task provided. The interface will provide access to explanatory details, allowing instantaneous evaluation of the driving factors behind the ranks allowing dispatch to determine, with confidence, which resource is best able to resolve a given task and prioritize

assignments accordingly. Current dispatch solutions in the marketplace ignore the optimization question, instead assuming that any resource in a group is equally capable at resolving a given task associated with that group.

Application Description

The application composes the resource rankings for each task by extracting activity descriptors from the text provided and combining it with existing knowledge about the customer, site and available resources. Using this process, on an internally hosted application, protects business critical proprietary information while maximizing the benefit gained from that information.

Data Description

To operate this pipeline, business critical, confidential, and proprietary information will need to be compiled. Installing this application will require populating the database with details on customer companies, sites and contacts as well as measuring the expertise of resources at various tasks and technologies. The quality of this data is critical to the proper functioning of the pipeline.

To prevent harm to any person or business, the demonstration dataset will be composed of randomly generated data curated to resemble real information, lacking the anomalies that would be present in real data. This dataset will be composed of textual identifiers and numeric weight values. Where practicable, repeating values will be stored separately and referenced as needed.

Objective and Hypothesis

Resulting from this project, a data processing pipeline will accept a formatted description of a requested task. It will rank the available resources on their ability to efficiently resolve the request. The attributes and weights underlying these rankings will be available for examination by system users. Operation of the pipeline is premised on the theory that if a service organization's collective knowledge about their customer's sites, customer's systems and available organizational resources were combined, then optimal resources could be efficiently and accurately matched to any task requested by those customers.

Methodology

The traditional waterfall project management methodology will be used to develop this product. This project management style is the most cost effective option for smaller projects in environments with fixed requirements over the life of the project.

The requirements phase of the waterfall method corresponds to the initial discussions and refinements of needed features for this project. This task will be accomplished using input from technical service industry experts. The design phase of the project follows requirements gathering and clarification. This will see the application architecture determined and data models defined. Configuration of the database, programming environment and coding of the application itself will occur in the implementation phase, after the conclusion of the design process. Once implementation completes, the functions, modules and application will be tested in the verification phase, ensuring proper behavior across both intended and unintended

inputs. The project will pass into the maintenance phase where any bugs missed in verification and any regulatory concerns can be resolved.

Funding Requirements

This application will operate on a dedicated, secure linux server, with an associated hardware cost of $3000.00 specified to accommodate widely varied workloads. It will require dedicated software engineer resources totalling to $24,000 over the course of 240 working hours at $100 per hour. The project will utilize development tools and software libraries available at no cost and with no associated licensing fees. Completing this project will cost $27,000 in total with expected quarterly maintenance averaging to $600 per year with $4800 per year in labor. Annual maintenance costs are expected to average to 20% of installed cost.

Stakeholders Impact

The application will allow a variety of practical efficiency gains for service resources, dispatch staff, customer representatives and service management. Service resources will benefit with better utilization of training and more applicable training. Allowing them to be more productive and experience more predictable work assignments. Dispatch staff will be able to more quickly evaluate tasks and select the right resources to bring a swift resolution. This opens opportunities to better track and examine trouble cases and encourages more complete task description. Customers representatives experience more consistent interactions and reduced disruptions as knowledgeable technical resources resolve problems faster and with fewer trips. Service management benefits from higher work quality, as resource proficiency is matched to tasks, reduced costs, as rework and return trips are reduced. As a system, these stakeholders

see decreased costs and increased quality as the existing workload is optimized, increasing work

quality and decreasing both length and quantity of visits.

Data Precautions

While the data that will be used in the installed application is sensitive, it is not necessarily bound by regulations in the United States. Constructing the project to minimize the duration a portion of the data is kept in memory and using a modular structure will facilitate changes required for future regulatory compliance. Existing regulatory frameworks, HIPAA and HITECH for healthcare, PCI DSS for credit card processing and FERPA for educational data, all have recommendations and requirements that encourage secure computing. These requirements vary in scope and specificity, ranging from requiring role-based access to mandatory password rotation on to encryption of data at rest and in transit. In support of easing compliance with future regulation efforts, the application will be divided into discrete modules, connecting to a shared, secure MariaDB database engine for storage and retrieval of sensitive information. Additionally, password information will be securely stored using verified hashing functionality.

Developer Expertise

This project will be completed by a degreed computer science engineer bringing more than 12 years of experience in the technical service industry; a wealth of knowledge and experience, supported by a variety of certifications, to the project.

The software engineer's experience with project management, using the waterfall method, is underpinned by both ITIL 3 Foundation certification and CompTIA Project +. Validating this

experience with graphical user interface design technologies is CIW Site Development Associate. A Microsoft Professional Certification in Data Science and Udacity Machine Learning Nanodegree demonstrate expertise in application of machine learning techniques. These academic and professional certifications and achievements provide a strong foundation for completing this project as scheduled and budgeted.

Prompt B

Project Proposal

PROBLEM STATEMENT

Interpreting problem descriptions in context and identifying the best available resource to resolve that problem is the challenge faced by dispatch teams at every service based company. Accurate analysis of the relevant knowledge an organization has and prediction of technical resources best suited to efficiently resolve an issue can provide tangible benefits to all stakeholders. These benefits can manifest as reductions in the number of tasks, hours required to resolve, trips required to resolve and frequency of rework.

CUSTOMER SUMMARY

This application is intended to serve dispatchers of service organizations. It's purpose is to supplement their knowledge and skill set with information gathered and compiled from throughout their organization. It will present a best-effort ranking, with explanatory descriptors and weights. This opportunity to review the driving factors provides these subject matter experts reminders of overlooked details and opportunities to correct faulty decisions and associated knowledge.

The task pipeline will be structured to be flexible in its deployment. The initial version will be targeted to run on a dedicated linux server and provide a browser-based interface on the local network. This application will assume users have basic web navigation skills.

EXISTING SYSTEM ANALYSIS

Currently, the customer, Seamus Company, uses a standard service ticket application. This tool allows a member of the dispatch team to open tickets upon receiving a request from customers via email or over the phone. The ticket, customer information and available resources are then reviewed by a member of the dispatch team. That team member chooses an available technical resource to dispatch to resolve the ticket.

At the completion of the project, a physical server will be installed, providing a dispatch support interface on the Seamus Company internal network. This server will use Ubuntu Server 18.04.4 LTS as its operating system. Python 3.7 will be installed with the following libraries: Bcrypt, Flask, Flask-Security-Too, Flask-WTF, Keyring, Matplotlib, Natural Language Processing Toolkit (NLTK), PyPubSub, SQLAlchemy, WTForms and Urllib. The final, intended result of this project is to provide direct, consistent and actionable information and recommendations to the dispatcher during their review of the ticket. If successful, this will improve outcomes for all stakeholders involved.

DATA

To operate this pipeline, business critical, confidential, and proprietary information will need to be compiled. Installing this application will require populating the database with details on customer companies, sites and contacts as well as measuring the expertise of resources at

various tasks and technologies. The quality of this data is critical to the proper functioning of the pipeline.

The demonstration dataset will be composed of generated data curated to resemble real information though lacking the anomalies that would be present in real data. This dataset will be composed of textual identifiers and numeric weight values. Where practicable, repeating values will be stored separately and referenced as needed.

As resource skill sets change, new customers and sites are added, new information about customers and sites becomes available or systems are replaced, updates to the data housed in this application will need to be applied. The application will include the interfaces required to perform these database updates.

PROJECT METHODOLOGY

Over the course of this project, planning and scheduling will follow the waterfall method. Using traditional waterfall project management will see the application development process pass through requirements gathering, design, implementation, testing and maintenance. This method, while less flexible than alternatives and less robust when coping with requirements changes, is cost effective and efficient on small projects and those instances where requirements are expected to remain stable.

Requirements gathering includes the initial discussions about the needs of dispatchers and concludes when agreement is reached on the feature set to be included in the application.

Design follows requirements gathering and clarification. This will see the application architecture determined and data models defined. These phases will be completed prior to

implementation of the application, when configuration of the database, programming environment and coding of the application itself will occur. Once implementation completes, the application will be tested in the verification phase, ensuring proper behavior across both intended and unintended inputs. As a result of this testing decision, extra time will be allotted during the testing phase to resolve bugs discovered. Finally, the project will pass into the maintenance phase where any bugs missed in verification and any regulatory concerns can be resolved.

PROJECT OUTCOMES

Upon completion of the project, the finished application will be delivered to the customer. The application will include a browser-based graphical user interface, providing access to user and administrative interfaces and allowing access to maintenance functionality. In addition, the finished application will include the demonstration database, and a demonstration user for each of the application access roles. For documentation purposes, the user guide will be provided, providing installation instructions and explaining the user interface. Finally, the project schedule with projected and actual milestone completion dates will be provided.

IMPLEMENTATION PLAN

Coding of the application will be completed in a top-down approach. This approach has been chosen to allow functionality verification of each module and to ensure that they interface correctly with the broker module. First, the data models will be defined, implemented, and populated with generated data. Second, the pipeline broker will be established, providing publisher-subscriber signaling between modules and encapsulating the PyPubSub library. Third,

a generic pipeline interface class will be established, providing generic pass-through functionality. Each of the required modules will be constructed as stubs, derived from the generic pipeline interface. Fourth, each of the modules will be filled out, replacing the stub functionality with the required processing. Fifth, the GUI will be constructed and attached to the completed pipeline.

Upon completion of the GUI, each test task will be added to the database. As the processing steps are completed, their outputs will be logged to the database for examination. When errors occur, the responsible module will be examined to locate and correct the faulting logic before the entire test process is restarted.

Once functionality has been verified, deployment of the application to the production server will follow a separate set of phases; environment setup, application installation and database population.

During the environment setup phase, Ubuntu Server 18.04.4 LTS will be installed and updated to the most current long-term support configuration. MariaDB will be installed and configured as the database engine. Python 3.7 will be installed, along with the dependency libraries, and NLTK will be configured. Application installation will require creation of the permanent directory for the application and configuration of the database connection details in pipeline.config. Launching the application once the configuration is in place will allow it to securely store the database connection details and remove the provided password from the config file. Finally, if the application is ready for deployment to production, the provided demonstration data must be removed and replaced with customer business data.

During the development process, all portions of the environment setup can be completed, though later deployment stages must wait until the development process is completed. Conversely, the development process has no dependencies within application deployment. Upon delivery of the configured, physical server, the customer will also receive the user guide and production schedule documentation. As extended maintenance is expected, documentation of repairs made, updates performed and enhancements applied will be provided after each maintenance event.

EVALUATION PLAN

The application will be validated using curated task assignments. These demonstration tasks have been selected as representative of normal service request tasks and assigned an ideal resource for grading purposes. The pipeline is said to have been successful if the ideal resource appears in the top 10% of rankings. In the case of the demonstration database, with 25 defined resources, success requires the ideal resource be in the top 2. Overall, the application will be considered successful and reliable for customer needs if it can achieve 70% success over all test tasks by the above metric. This criteria will bias the application towards the desired result, matching technical resource proficiency to customer, site and task needs.

This pipeline acts as a tool to provide assistance to existing human dispatchers, using information already available to the organization. Concentrating it and analyzing it in this way does not create new liabilities under existing United States regulatory frameworks. Nor does it interact meaningfully with existing industry standards.

RESOURCES AND COSTS

For this application, a single server will be required, with the natural fluctuations of server prices, it will cost approximately $2,500. Taxes and shipping are expected to add as much as $500 to the final price of $3,000. The libraries and applications used in this project are available for use without additional licensing costs. Final delivery is anticipated to require 8 hours of labor, costing $800, as accounted for in the initial project schedule and budget.

TIMELINE AND MILESTONES

Development and deployment is anticipated to require 240 hours of labor over the course of 13 weeks. This schedule is a result of outside constraints on the schedule of the assigned software engineer. Completion of this project is dependent on the milestones described below.

Mile-stone	Pre-requisites	Activity	Resource Assigned	Hours	Start	End
1	-	Requirements approval	Project Manager	8	5/4/20	5/5/20
2	1	Architecture design	Software Engineer	16	5/6/20	5/11/20
3	1	Database design	Database Engineer	16	5/12/20	5/15/20
4	2, 3	Server purchase	Procurement Specialist	4	5/18/20	5/18/20
5	4	Server receipt	Procurement Specialist	4	6/18/20	6/18/20
6	5	Server environment setup	Software Engineer	8	6/19/20	6/22/20
7	2, 3	Development environment configuration	Software Engineer	8	5/20/20	5/21/20
8	7	Database creation	Database Engineer	8	5/22/20	5/26/20
9	8	Data generation and curation	Software Engineer	16	5/27/20	6/1/200
10	7	Pipeline broker creation	Software Engineer	8	6/2/20	6/3/20
11	7	Generic pipeline interface	Software Engineer	12	6/4/20	6/8/20
12	11	Pipeline parser module	Software Engineer	12	6/9/20	6/11/20
13	11	Pipeline tagger module	Software Engineer	12	6/12/20	6/16/20
14	11	Pipeline ranker module	Software Engineer	12	6/17/20	6/25/20
15	7	Pipeline GUI	Software Engineer	20	6/26/20	7/2/20
16	10, 11, 12, 13, 14	System testing	Quality Assurance	40	7/6/20	7/17/20
17	16	Application deployment	Software Engineer	8	7/20/20	7/21/20
18	17	Application verification	Software Engineer	20	7/22/20	7/28/20
19	18	Final project delivery	Software Engineer	8	7/29/20	7/30/20

Prompt C

Application Files

\task_pipeline

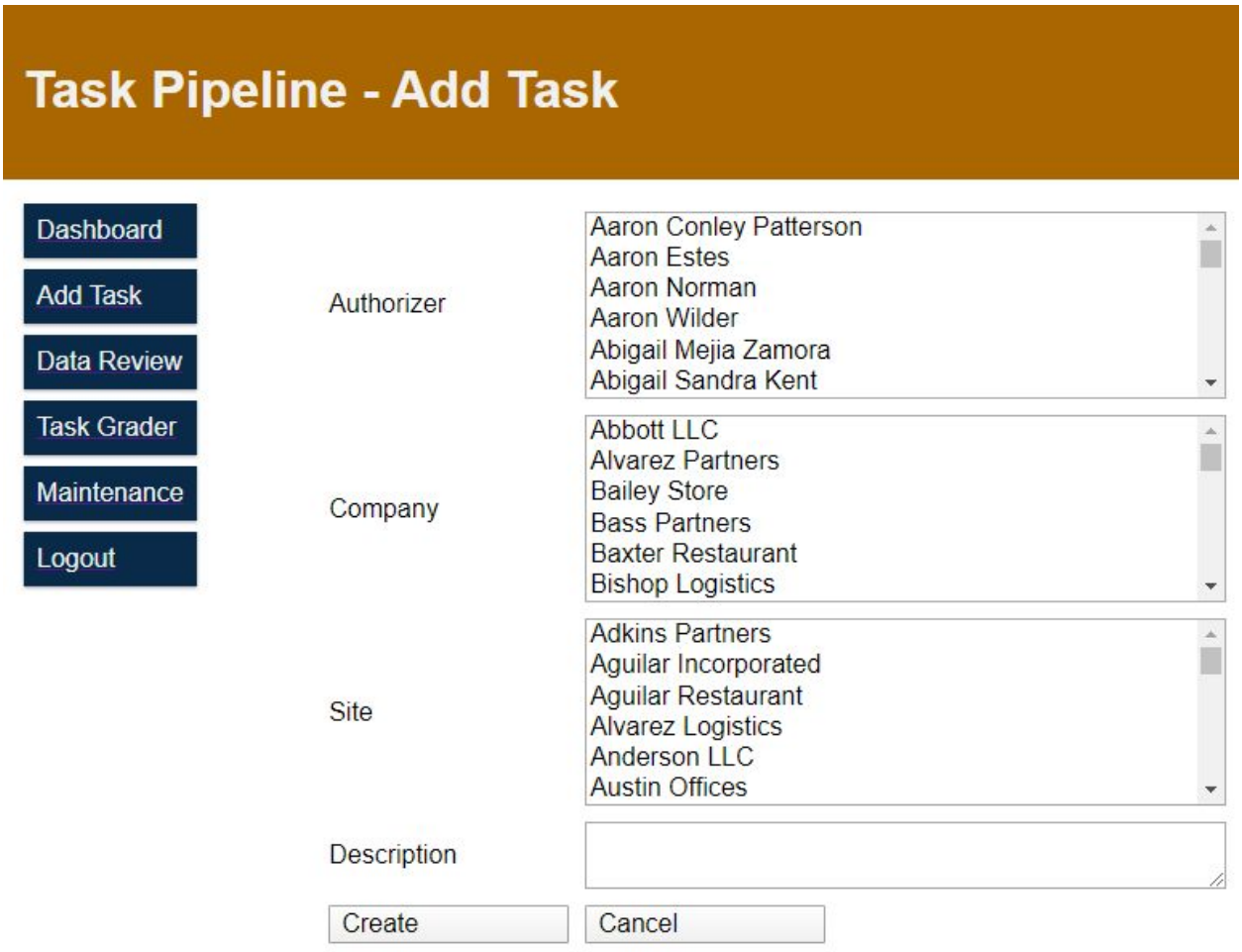
\config	
demo_database.sql	SQL script creates the db and inserts demo data
database_string.config	Configuration file used by the data generator
pipeline.config	Configuration file used by the application
synonym_map.json	Listing of attribute associations and synonyms
task_sample.json	Listing of tasks used for verifying the pipeline
\generators	Generator files used to initially generate demo data
\static\pipeline.css	CSS file used throughout the GUI for consistency
\templates*	Template files for each page within the GUI
\dependencies	Listing of required python libraries
\pipeline_app.py	Application initializer
\pipeline_database.py	Database connection initializer
\pipeline_form.py	HTML form definitions for the GUI
\pipeline_helper.py	Miscellaneous static utility functions
\pipeline_model.py	Database model definitions for SQLAlchemy
\pipeline_service.py	Broker and pipeline module definitions
\pipeline_view.py	Flask routing and user interaction handling
\app.py	Application launcher.

Prompt D

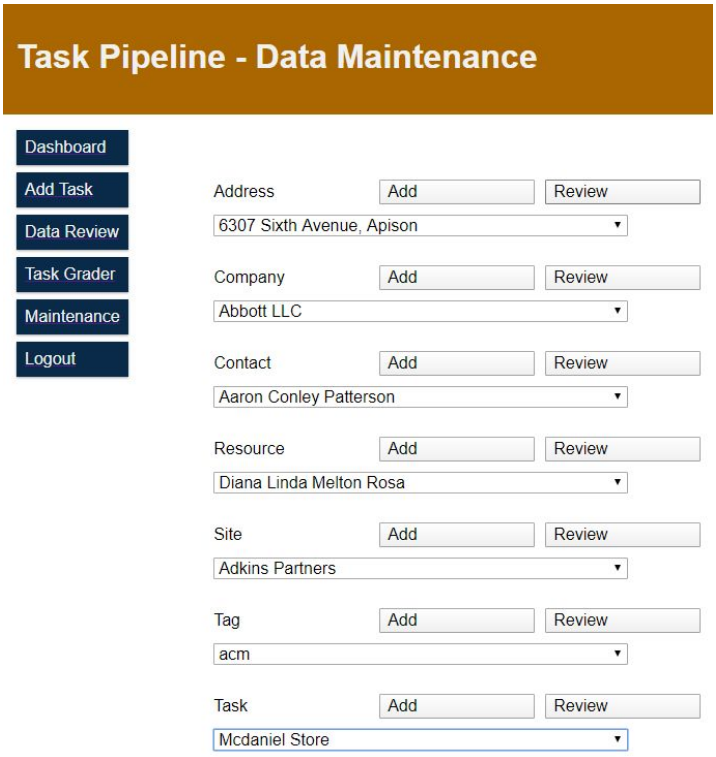
Post-implementation Report

Project purpose

The task pipeline project provides decision-support to service dispatch staff by using a database of information relating customers, sites and resources to predict which resources would be best suited to resolve a given service task efficiently. By comparing weighted tags of technical resources to generated tag weights for service tasks, the application attempts to rank those



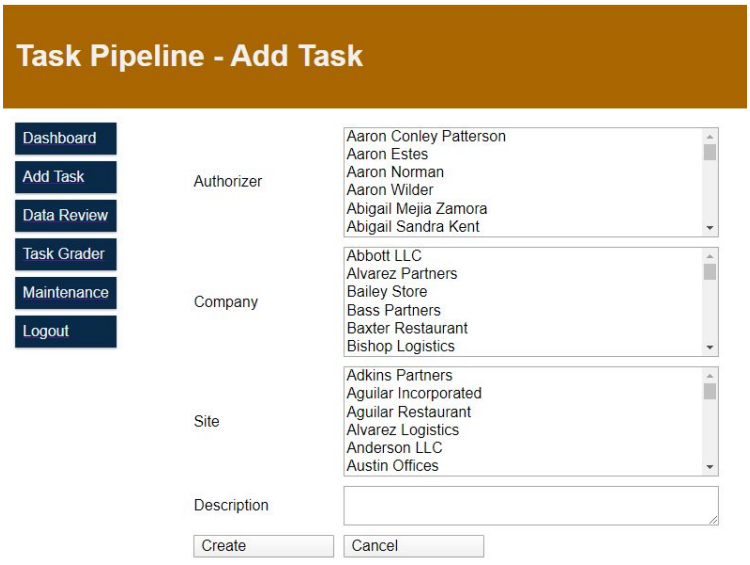
resources by estimated proficiency in performing the service requested. By performing these estimations and providing an interface to examine the component estimates, the application provides the user with high confidence in the system and enables them to review and reconcile rankings with which they have cause to disagree.



Datasets

The sensitive nature of the data required for effective operation of the task pipeline precluded its use in the demonstration database. As a result, data was generated using generic data with the aim to recreate the look and feel of curated, real-world data.

This day has been stored in 3rd normal form, with the exception of address data, for which all fields are typed data entry, with no normalization. This format was chosen to reduce the depth of the joins required to form complete contact, company, site and resource records.



Data product code

In an effort to facility future modifications and to reduce exposure to systemic bugs, the task analysis process was decomposed into discrete, disconnected steps. The steps, and therefore modules, included in the pipeline are text parsing, text tagging and resource ranking. Each module of the task pipeline receives the task id of the next task it should process from the broker. The module then retrieves the current status of that task from the secure database, performs its processing step and writes the result back to the database. After the database writing task is complete, the module sends the task id back to the broker, signalling that it has completed its work on that task.

The text parser divides the task description into discrete sentences before extracting the nouns from each sentence. These nouns are used by the text tagger as tags with a neutral weight, providing an abbreviated, descriptive view of the task. The resource ranker combines the noun-based tags with existing knowledge about the site, company and, if available, customer contact and manually added task tags to create a single composite weight for each tag included in any contributing entity. This composite weighted tag list is then compared against the tags assigned to each available resource. Each resource receives a score estimating how closely matched their skill set is to the needs of the task. When sorted in ascending order, by score, this listing becomes a predictive ranking of the most efficient resource to assign to the task.

Hypothesis verification

The design of this application was informed by the hypothesis that if a service organization's collective knowledge about their customer's sites, customer's systems and available organizational resources were combined, then optimal resources could be efficiently and accurately matched to any task requested by those customers. Using the randomly generated dataset, the hypothesis could not be confirmed. However, the behaviour of the system and performance levels that were reached suggest that the hypothesis cannot yet be rejected. Improvements to any part of the pipeline, or to the generator algorithm, could show significant improvements in the performance of the pipeline.

Effective visualizations and reporting

As a decision-support application for dispatch, the visualization of primary importance is the representation of the tasks to be dispatched and the ranked list of available resources. In this regard, from the dashboard, the user is able to quickly assess which sites have tasks requiring dispatch and complete those dispatch activities.

From the same dashboard, the frequency with which certain resources are ranked highly is immediately accessible as well as which tags are receiving the most activity. The data review section of the interface allows access to the underlying data for addition and modification. To protect referential integrity in the database, records can be set to inactive through the interface but cannot be deleted.

Accuracy analysis

For the accuracy metric for this application, it was determined that a recommendation would only be considered successful if the ideal resource for a given task was in the top 10% of rankings for that task. With a field of 25 generated technical resources, this requires that the ideal resource be one of the top 2. As can be seen on the dashboard, the pipeline achieves that level of success 48.3% of the time across 29 tasks. By this predetermined metric, on randomly generated data, the pipeline fails to perform at reasonable levels. Review of the data and predictive behavior suggests that greater differentiation between technical resources would improve pipeline performance dramatically.

Application testing

Using a top-down development methodology allowed the functionality of each module to be tested as part of the integrated whole throughout its coding. This resulted in the input and output of each module being inspected at the database level to ensure the expected results were achieved. Each modification to a module required resetting the demonstration database and re-executing the pipeline. This proved to be more labor intensive than implementing unit testing. The resulting modular pipeline has been confirmed to behave as intended with each module accepting the expected inputs and providing the expected outputs.

Appendices

Installation Guide

Prerequisites:

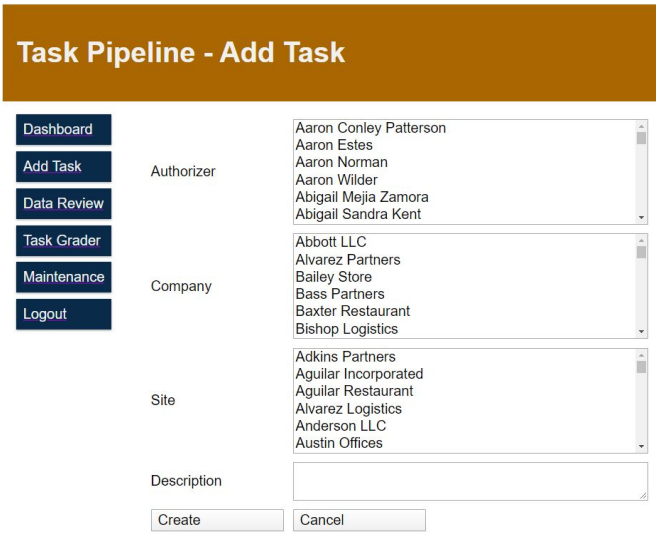
MariaDB Server Instance

Python 3.7 with supporting libraries: Bcrypt, Flask, Flask-Security-Too, Flask-WTF, Keyring, Matplotlib, Natural Language Processing Toolkit (NLTK), PyPubSub, SQLAlchemy, WTForms and Urllib.

1) Install the prerequisite applications.
2) Extract task_pipeline.zip into the directory from which the application will run.
3) Create an account on the database server instance with db_owner and db_creator permissions.
4) If the demonstration database will be used, run /task_pipeline/config/pipeline_data.sql
5) If not, create an empty database, granting db_owner to the account created in step 3.
6) Edit /task_pipeline/config/pipeline.config, inserting the connection information for the installed database.
7) Run python /task_pipeline/app.py
8) If the application is installed on a workstation, open a web browser and type http://127.0.0.1:5000/

9) If the application is installed on a server, from any local workstation, open a web browser to http://<server address>:5000
10) Demonstration user accounts exist for each permission level. These accounts should be changed on first application launch.
    a) admin user: admin@example.com password:
    b) dispatch user: dispatch@example.com password:
    c) user user: user@example.com
11) For production use, data entry is available through the Data Review page.

User Guide



To start task processing, configure a new task through the add task interface. Once the task is created, it will be passed through the pipeline automatically.

For administrators, processed tasks will appear in the Ungraded Tasks interface of the Task Grader. To activate the task grader, select the task and resource, then click Assign Ideal Resource. Once this is completed, the grader will determine whether the pipeline was correct or incorrect and assign the appropriate score.

For dispatch users, once the pipeline has finished processing the task, it will appear in Tasks awaiting dispatch on the Task Pipeline Health Dashboard. Selecting any task and clicking Dispatch will open the dispatch interface for that task.

Summation of Learning Experience

In approaching this assignment, I was again confronted with my own issues with scale and troubles with vague directions. My educational experiences to this point and desire to be productive with my time encouraged me to choose a task that would solve an issue I encounter in my daily life. Learning to build graphical interfaces in Java gave me a grounding in graphical interfaces I previously lacked. Learning project management skills and gaining 2 certifications to validate those skills also put me in good stead to understand the processes involved. Finally, building multiple applications from scratch, resulting in functional interfaces, gave me the confidence to approach a more serious developmental challenge.

I chose to build this application in python, a choice that required I learn python-specific graphical interface technologies. For this, I chose Flask, a simpler framework that would support the project development. Learning the flask ecosystem also introduced me to SQLAlchemy, providing a significantly improved SQL interface compared to the encapsulated database connector I had built. Learning these additional libraries and techniques from the documentation was a challenging and enlightening experience. Aside from checking in with my course instructor and program mentor every week or two, this work was completed without assistance. This project has encouraged me to continue expanding my repertoire programming knowledge and to seek certification of those skills.

References

This space intentionally left blank.