Team Bio

❖ Nick Cason
  ➢ Systems Administrator

❖ Andrew Davidson
  ➢ Software Engineer

❖ Khang Nguyen
  ➢ Systems Administrator
  ➢ VDI Engineer

❖ Josh Pena
  ➢ Non-contributor

❖ Sean Tosloskie
  ➢ Sr. Developer
  ➢ Soccer player

❖ Alex Tucker, RHCSA
  ➢ Systems Engineer

❖ CAPT Mark “Forrest” Stoops, USN
  ➢ Director, C4I & Space
# Table Of Contents

I. Problem Background .............................................. 4  
II. Problem Statement ............................................. 5  
III. Problem Characteristics .................................... 7  
IV. Customer/End user ............................................. 9  
V. Current Process Flow ......................................... 10  
VI. Solution Statement ........................................... 11  
VII. Proposed Process Flow ..................................... 12  
VIII. Major Functional Components Diagram ............... 13  
IX. Competition Analysis ....................................... 14  
X. Build Tools ...................................................... 15  
XI. Core Components ............................................. 17  
XII. User Interface ................................................ 18  
XIII. Infrastructure ............................................... 25  
XIV. Backend ....................................................... 40  
XV. Risks: Overview .............................................. 48  
XVI. Technical Risks .............................................. 49  
XVII. User Risks .................................................... 56  
XVIII. Customer Risks ............................................ 60  
XIX. Agile Development ......................................... 64  
XX. Conclusion ..................................................... 65  
XXI. References .................................................... 66  
XXII. Glossary ....................................................... 69
Problem Background

“We get stacks of documents as thick as phone books ... most don’t read past the executive summary.”
- Capt. Mark Stoops, USN

❖ Naval intelligence produces more information than can be consumed in a reasonable timeframe.
❖ Finding significant connections between new and existing intelligence documents is a difficult problem.
❖ Meet our mentor on page 3 of the handout
“Documents deliver *information*. The links and relationships between them contain *meaning*. Curation of these relationships offers *clarity*.”
Documeta is a document insight & comprehension tool.

Visit us at our [website](#) for more information.
Problem Characteristics

❖ The relationships connecting documents and their information aren’t always apparent
❖ Relationships are inferred by the reader
  ➢ Unnoticed connections
  ➢ Oversights, inattentiveness, incompetence, bias
❖ Determining relationships can require tedious effort
  ➢ Error-prone
  ➢ Frustration-prone
Problem Characteristics (continued)

- Large sets of documents: additional challenges
  - Time constraints
    - Takes considerably longer to find connections
    - Humans can only process one document at a time
    - Not feasible to find all possible connections
  - Daunting task
    - "Where should I begin my search? This is over 1,000 documents"

- Important questions need quick answers
  - There isn’t always time available to read through documents
  - "I don’t have time to read."
Who is our specific application relevant to?

❖ Our Customer: Capt. Mark Stoops of the U.S. Navy
❖ Our End Users: Enlisted members of the U.S. Navy
❖ The general case:
  ➢ Project Managers
  ➢ Researchers
  ➢ Consultants
Current Process Flow

- Inefficient
- Time Consuming
- Wasted Effort
Solution Statement

“Extracting important characteristics from documents reveals relevant information on a topic. Visualizing these discoveries in a fun, interactive, and meaningful way acts as a vehicle of tangential learning.”
Proposed Process Flow

**Frontend**

A user seeks to research and understand a domain topic

A collection of documents are provided to documeta

A listing of keywords is provided

Document breakdown and keyword relationships are visualized in an explorable, interactive interface

**Backend**

Process and normalize collection for uniformity

Normalized documents and keywords added to database

Are all documents processed?

Yes

No

Parse and extract metadata keywords from document

Calculate relatedness strength between document and keywords

Map relations ups among the set of document based on keywords

Document references are placed in respective categories in database
Major Functional Components refer to page 5 of the handout
## Competition Analysis

<table>
<thead>
<tr>
<th>Competitive Analysis</th>
<th>documeta</th>
<th>Tableau</th>
<th>Envisioning IO</th>
<th>Qlik</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Extraction</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Data Analysis</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Associative Indexing</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relational Visualization</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Visual Navigation</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Containerization</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
Build Tools

Technology Stack
❖ Django Web Framework
❖ MySQL Community Edition

Deployment Stack
❖ Kubernetes Container-Orchestration System
❖ Docker Software

User Interface
❖ Data Driven Documents (D3.js)

Document Text Extraction
❖ Textract

RESTful API
❖ Flask (Python web-framework)

Version Control
❖ Git
Core Components

- documeta
  - Fontend
    - Bootstrap v5
    - D3 Framework
  - Infrastructure
    - Kubernetes
  - Backend
    - Flask
    - Textract
  - HTML
  - CSS
  - SVG
  - Traefik
  - Docker
  - RESTful API
  - HaProxy
  - Nginx
  - MemCache
  - MySql
User Interface

Data Driven Documents
D3.js library

- Javascript library for producing dynamic, interactive data visualizations in web browsers
- Embedded within an HTML webpage
- Uses pre-built Javascript functions to:
  - Select elements
  - Create & style SVG objects
  - Add transitions, dynamic effects, or tooltips to SVG objects
- Large datasets are bound to SVG objects using D3.js functions to generate rich text/graphics charts and diagrams
- Binds arbitrary data to a Document Object Model (DOM)
D3.js

- Built upon a series of injectors, or an *injection chain*
- There are four main components that play a part in the processing chain of D3.js:
  - Input Data
  - Element binding
  - Method chain
  - DOM *(Document Object Model)*
D3.js | Technical Principles

Selections:
- Enables a programmer to select a given set of DOM nodes, and use operators to manipulate them.
- Selection can be based on:
  - Tag
  - Class
  - Identifier
  - Attribute
  - Place in the hierarchy
- Operations include:
  - Accessing/mutating attributes
  - Adding/removing attributes
  - Display text
  - Display styles
- All operations to HTML elements can be made dependent on data.
Transitions:

❖ Transitions are a form of animation
❖ A D3 transition is a “selection-like” interface for animating changes to the DOM
❖ Will smoothly interpolate the DOM from its current state to a desired target state over a given duration
❖ D3 transitions will determine an appropriate interpolator by inferring a type including:
  ➢ Numbers
  ➢ Colors
  ➢ Geometric transforms
  ➢ Strings with embedded numbers (e.g., “100px”)
D3.js | Technical Principles

Data-binding:
- Loaded data can drive the creation of elements
- D3.js loads a given dataset, then for each of its elements can:
  - Create an SVG object with associated properties
    - Shape
    - Color
    - Values
  - Create an SVG object with associated behaviors
    - Transitions
    - Events

Example code snippet:
```javascript
var svg = d3.select("#hook").append("svg")
    .attr("width", 120)
    .attr("height", 120);

var data = d3.json("missiles.json");

svg.selectAll("circle")
    .data(data)
    .enter()
    .append("circle")
    .attr("id", function(d) { return d.name })
    .attr("cx", function(d) { return d.range / 1000 })
    .attr("cy", function(d) { return d.speed })
    .attr("r", function(d) { return d.count / 1000 * 2 })
    .attr("fill", function(d) { return d.color });
```

`missiles.json` content:
```
{
  "name": "Missiles",
  "children": [
    {
      "name": "PL-8", "range": 53000, "speed": 780, "count": 6378, "color": "black"},
    {
      "name": "PL-9", "range": 73000, "speed": 870, "count": 5094, "color": "blue"},
    {
      "name": "PL-300", "range": 27000, "speed": 580, "count": 3407, "color": "grey"
    }
  ]
}
```
User Interface: Proposed Model

Refer to page 3 of handout for additional information on the D3 API
Infrastructure

Modern Deployments
Infrastructure Overview refer to pages 7 & 8 for brief overview of select technologies

Docker & Kubernetes
- Scalability
- Resiliency
- Portability
- Cloud Ready

MySQL & Memcached
- Enterprise Support
- Native JSON data types
- Encryption & Security
- Performance

Nginx
- Async, Event-driven
- Concurrency
- Caching, load-balancing
Infrastructure Overview *cont.*

**Kerberos & LDAP**
- Security
- Reliability
- Scalable

**Traefik & Haproxy**
- HTTPS load-balancing
- Ingress
- Internal load-balancing

**Gitlab**
- Continuous Integration
- Continuous Deployment
- Issue Tracker
Docker

- Patching and Updates
- Scaling
- Deployment
- Cloud Ready
Kubernetes

- Management
- Automation
- Integration
Nginx

- **Performance**
  - Smaller footprint
  - Modular
  - Event based

- **Enterprise Options**
  - Support
  - More features
  - Active Development
Memcached

- **Performance**
  - UI Experience
  - Scaling
  - Improved utilization
OpenLDAP

- Directory Services
- Internal to each Pod
- Identifies service UPNs
Bind9

- DDNS
- Internal Resource Identification
- isc-dhcp-server companion
MIT Kerberos KDC

- Authentication
- Authorization
- Encrypt Credentials over the Network
Traefik

- Ingress ACLs
- Load Balancing
Proposed Design

Kubernetes

- Traefik
- Application Pods
- Stateful replicating cluster
Security Considerations

- Pod Isolation
- Internal API end-to-end Encryption
- Ingress & ACLs
- Handling krb5 Keytabs in Containers
Continuous Integration / Deployment

❖ Gitlab

➢ Continuous Integration
➢ Continuous Deployment
➢ Issue Tracker
➢ Enterprise Support Options
Backend

Automated Document Classification
Backend Overview

- Backend
  - Text Extraction
  - Metadata Extraction

- RESTful API

- User Interface

- Database
A collection of documents serve as an argument for the backend
- Collection of various formats (pdf, docx, etc)

Documents are normalized into a uniform text format

Utilizing Textract, a Python text extraction library
- Wrapper around several different extraction libraries
- Supported document formats include: .pdf, .docx, .html, .xlsx, .pptx, and many more
- Simple interface:

```python
import textract
text = textract.process(file).decode('utf-8')
```
Metadata Extraction: Goals

❖ Automate the classification of each document
  ➢ Determine the category of focus for each document
❖ Create a structure the frontend can process efficiently
  ➢ Communicate with frontend via RESTful API
Metadata Extraction: Implementation

❖ Extract the metadata
  ➢ Break text up into sentences
    ■ Offers more efficient means for locating keywords
      ○ concurrent regex operations over a set of sentences
  ➢ Extract the relevant information
  ➢ Sentences containing keywords will be used for relaying information user desires
    ■ only display the relevant information

❖ Calculate the relatedness strength of target keywords
  ➢ As sentences are parsed, record a reference count of explicitly mentioned keywords
    ■ Target categories are read in from file (yaml), placed into fast access structure
    ■ Text is broken up into individual words
    ■ Words are matched against target categories

❖ Processed metadata is stored in database
  ➢ Avoid reprocessing documents every request
Database | MySQL RDBMS

- **dbo.Documents**
  - Contains all uploaded documents

- **dbo.Keywords**
  - All predefined keywords used for text processing

- **dbo.Frequencies**
  - Determine the size of categories displayed in the U.I.
  - Number of matches made for given keyword during text processing
  - doc_id and key_id are attributed to each matched keyword
    - establish relationship between document and keyword
The extracted metadata is ultimately serialized in a JSON object the frontend will process

- Constructs the UI

Acts as a hierarchical structure

- Establish relationships between defined keywords and document entities

```json
{
  "name": "Order of Battle",
  "children": [
    {
      "name": "Ship",
      "children": [
        {
          "name": "Aircraft Carrier",
          "children": [
            {
              "name": "LIAONING CV-16",
              "children": [
                {
                  "name": "General",
                  "children": [
                    {
                      "name": "Hull Number",
                      "size": "n"
                    },
                    {
                      "name": "Power Source",
                      "size": "n"
                    },
                    {
                      "name": "Propulsion",
                      "size": "n"
                    },
                    {
                      "name": "Speed",
                      "size": "n"
                    },
                    {
                      "name": "Crew Size",
                      "size": "n"
                    },
                    {
                      "name": "Mission Areas",
                      "children": [
                        {
                          "name": "Surface Warfare",
                          "size": "n"
                        },
                        {
                          "name": "Anti-Submarine",
                          "size": "n"
                        },
                        {
                          "name": "Surveillance",
                          "size": "n"
                        },
                        {
                          "name": "Command & Control",
                          "size": "n"
                        }
                      ]
                    }
                ]
            }
          ]
        }
      ]
    }
  ]
}
```
Communication with Frontend

❖ Constructed JSON object is sent to frontend for U.I.
➢ Assemble and update

❖ Communication is handled through a REST API built with the Flask web-framework
➢ GET
   ■ User sends server request for updates
   ■ https://localhost:8080/endpoint/update
➢ PUT
   ■ Creation or update of JSON object
   ■ https://localhost:8080/endpoint/write
➢ DELETE
   ■ Removal of erroneous document references from U.I configuration
Risks

<table>
<thead>
<tr>
<th>Impact</th>
<th>Probability</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
<td></td>
<td>T1, C2</td>
<td>U1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T4, C1</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td>U3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Green: Low
Yellow: Medium
Red: Severe
T: Technical
U: User
C: Customer
Technical Risks

T1  Inaccurate Backend Results
T2  Erroneous Documents
T3  Database Failure
T4  Inability to Scale Performance
T5  Software Bugs
T6  Docker Failure
**T1: Inaccurate Backend Results**

**Description**
- Backend fails to place documents in correct categories (incorrect mappings)

**Mitigation**
- Recalibrate connections and rebuild database
- Explore alternative text extraction library

<table>
<thead>
<tr>
<th>Risk Matrix</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
</tr>
<tr>
<td></td>
<td>T1, C2</td>
</tr>
<tr>
<td></td>
<td>U1</td>
</tr>
<tr>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>T4, C1</td>
</tr>
<tr>
<td>Low</td>
<td>U2</td>
</tr>
<tr>
<td></td>
<td>C3</td>
</tr>
<tr>
<td></td>
<td>T2</td>
</tr>
<tr>
<td>Very Low</td>
<td>U3</td>
</tr>
</tbody>
</table>

410 Crystal 50 | Design
T2: Erroneous Documents

Description
❖ Malformed or corrupted documents are inadvertently added to document repository

Mitigation
❖ Have backend gracefully skip over any malformed or corrupted documents during processing
T3: Database Issues

Description
❖ Lack of documentation
❖ Corrupt logs
❖ Duplicate entries

Mitigation
❖ Concise documentation
❖ Log backups
❖ Data entry validation

Risk Matrix

<table>
<thead>
<tr>
<th>Impact</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
<td>T1, C2</td>
<td>U1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td>U3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
T4: Inability to Scale Performance

Description

❖ As the size of document sets increase, application performance becomes unacceptable

Mitigation

❖ Only process documents that are new or modified
❖ Perform periodic document processing at a user specified interval and time
❖ Memcache
# T5: Software Bugs

**Description**
- Serious bugs are introduced to code base during development, posing risk for application instability

**Mitigation**
- Test driven design
- Unit testing
- UI testing
- Continuous integration

<table>
<thead>
<tr>
<th>Risk Matrix</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
</tr>
<tr>
<td>High</td>
<td>T1, C2</td>
</tr>
<tr>
<td>Medium</td>
<td>U1</td>
</tr>
<tr>
<td>Low</td>
<td>T4, C1</td>
</tr>
<tr>
<td>Very Low</td>
<td>U2</td>
</tr>
<tr>
<td>Low</td>
<td>C3</td>
</tr>
<tr>
<td>Very Low</td>
<td>U3</td>
</tr>
</tbody>
</table>

410 Crystal 54 | Design
T6: Docker Failure

**Description**
- Host machine inadvertently loses communication with Docker daemon

**Mitigation**
- Restart the Docker daemon
- Run the Docker daemon in debug mode
- Audit the Docker daemon configuration file

<table>
<thead>
<tr>
<th>Risk Matrix</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
</tr>
<tr>
<td>High</td>
<td>T1, C2</td>
</tr>
<tr>
<td>Medium</td>
<td>U1</td>
</tr>
<tr>
<td>Low</td>
<td>T4, C1</td>
</tr>
<tr>
<td>Very Low</td>
<td>U2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Probability</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td>T4, C1</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>U3</td>
<td>U3</td>
<td>U3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td>U3</td>
<td>U3</td>
<td>U3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
User Risks

U1  Difficult Interface Navigation
U2  Erroneous Document Discovery
U3  Using Internet Explorer
### U1: Difficult Interface Navigation

#### Description
- User has difficulty navigating the user-interface

#### Mitigation
- Provide excellent user-documentation
- UX quality assurance
- Intuitive design strategy

---

#### Risk Matrix

<table>
<thead>
<tr>
<th>Impact</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
</tr>
<tr>
<td>High</td>
<td>U2</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
</tr>
</tbody>
</table>

---

#### Probability

<table>
<thead>
<tr>
<th>Impact</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
<td>T1, C2</td>
<td>U1</td>
<td>T4, C1</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### U2: Erroneous Document Discovery

**Description**
- User discovers an erroneous document while navigating the user interface

**Mitigation**
- Implement feature to allow user to flag inappropriate documents for removal

![Risk Matrix](#)

<table>
<thead>
<tr>
<th>Risk Matrix</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
</tr>
<tr>
<td>High</td>
<td>T1, C2</td>
</tr>
<tr>
<td>Medium</td>
<td>U1</td>
</tr>
<tr>
<td>Low</td>
<td>T4, C1</td>
</tr>
<tr>
<td>Very Low</td>
<td>U2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Impact</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Low</td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td>U3</td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

410 Crystal 58 | Design
U3: Using Internet Explorer

Description
- User is currently using Internet Explorer (i.e. a browser that is notoriously broken)

Mitigation
- Have application check if user is running Internet Explorer to display a gentle, disableable message recommending installation of Firefox or Chrome for the best UX
Customer Risk

C1  DoD Restrictions
C2  Inefficient Hardware
C3  Unsatisfied with Application
C1: DoD Restrictions

Description
❖ DoD compliance regulations prevent customer from installing application on DoD computers

Mitigation
❖ Maintain clear and consistent communication with customer to take necessary steps to make software DoD compliant
C2: Inefficient Hardware

Description
❖ Customer is currently using computers with hardware that does not meet the application minimum system requirements

Mitigation
❖ Customer explores cost effective hardware to run application
❖ Clearly define minimum hardware requirements

<table>
<thead>
<tr>
<th>Impact</th>
<th>Probability</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
<td>T1, C2</td>
<td>U1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T4, C1</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td>C3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td>U2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U3</td>
<td></td>
</tr>
</tbody>
</table>
C3: Unsatisfied with Application

**Description**
- Customer is unhappy with the application UI/UX

**Mitigation**
- Continuous Prototyping
- Iterative Design Strategy

### Risk Matrix

<table>
<thead>
<tr>
<th>Impact</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td>T3, T6, T5</td>
<td>T1, C2</td>
<td>U1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td>T4, C1</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>U2</td>
<td>C3</td>
<td>T2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td>U3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Agile Development

Agile Lifecycle

DEVELOPMENT PHASES
(Add Functionality)

REVIEW

FEEDBACK

APPROVE?

RELEASE

INTegrate & Test

NEXT ITERATION
(On to development & testing)

ADJUST & TRACK
(Reprioritize features)

RECORD & MAKE CHANGES

START

INITIATE PROJECT

Define Requirements

410 Crystal 64 | Design
Conclusion

- A document insight & comprehension tool
- Provides a visual interface to explore relationships between documents
- Organize and optimize information consumption
References


Glossary

ACL: Access Control List, a list specifies to the operating system the access and respective operations a user has towards a listed object.

Agile: A software development methodology focused on incremental, iterative development cycles; where requirements and solutions evolve through collaborations with customers via continuous prototyping.

API: Application Programming Interface, an abstract set of functions, communication protocols, and other various tools that specify how software components can interact.

Async: Asynchronous processing, units of work running separately from the main thread (process is not blocked on async operation).

Containers: A standard unit of software that packages up all code and dependencies into a sandboxed, virtual environment.

DNS: Domain Name System, a decentralized naming system used to associate various information with assigned domain names.

DoD: Department of Defense.

DOM: Document Object Model.

JSON: JavaScript Object Notation.

KRB5: Kerberos 5 (current release), computer network authentication protocol that works on the notion of tickets (allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner).

Method Chain: Technique used in Object Oriented Programming where multiple methods are invoked in sequence, with each method operating on the object returned by the previous call.

RDBMS: Relational Database Management System.

REST: Representational State Transfer.

Tangential Learning: A process by which people educate themselves on a topic when it is presented in a context that they enjoy (e.g. video games that teach educational topics).

UI: User Interface.

UX: User Experience.
Thank you all.