# Graphalyzer:

## A graph visualization and analysis tool

May1618

# May1618

Team Members:

- Andrew Bowler
  - SE
  - Webmaster
- Alberto Gomez-Estrada
  - CPR E
  - Communications
- Richard White
  - SE
  - Key Concept Holder
- Michael Sgroi
  - CPR E
  - Key Concept Holder
- Taylor Welter
  - SE
  - Team Leader
- Client: Workiva
- Advisor: Dr. Simanta Mitra

# Problem Statement

- Create a web application that visualizes Big Data into an interactive graph
  - Nature of data is arbitrary
- Perform graphical analysis on data
  - Depth, breadth, interconnectedness

# Goals of Graphalyzer

- Simplify data analytics through interactivity
  - Make it easy and intuitive to distinguish important relationships
  - Use basic visual attributes such as color and shapes
- Keep everything general
  - Be able to perform said analysis on any type of data
  - Eg. Payroll system for a large company

# Backend Functional Requirements

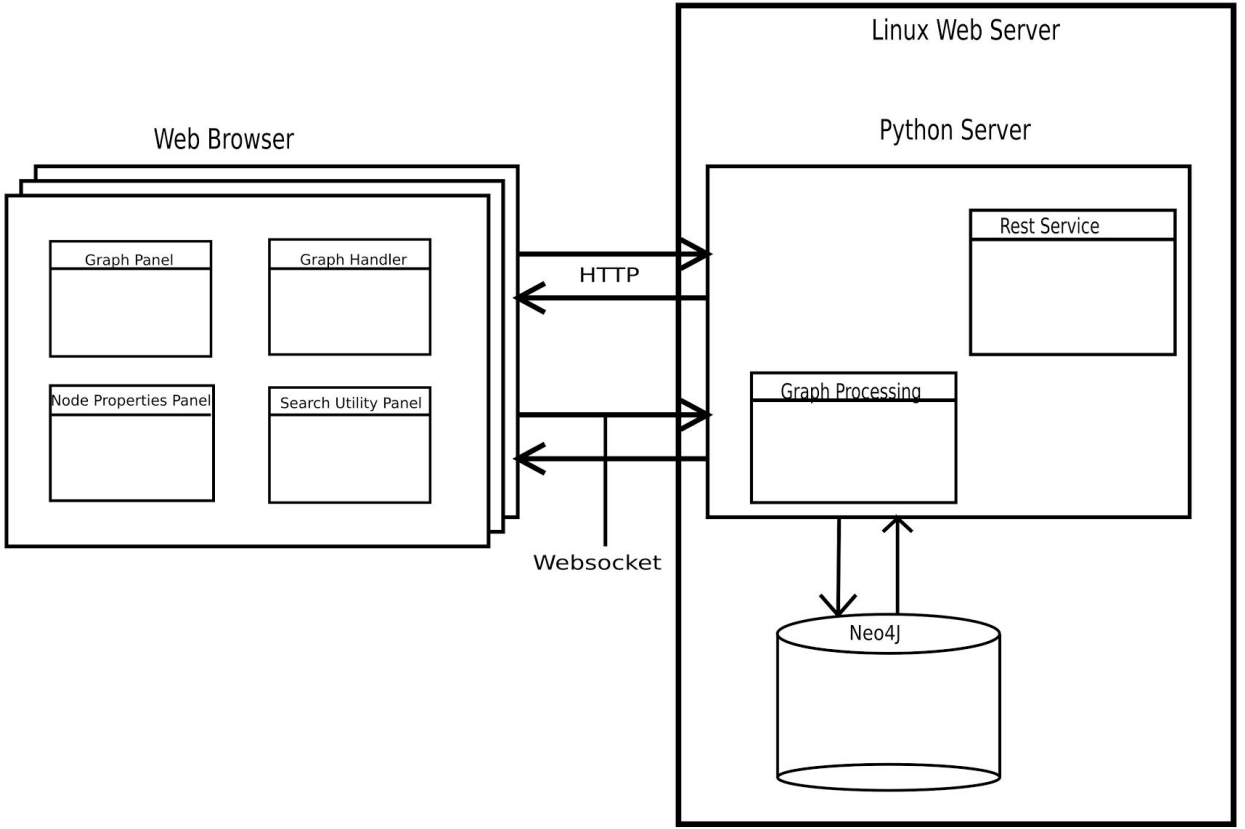
- Upload graph files of arbitrary data via REST
- Index incoming information associated with nodes and edges
- Allow for searching of individual nodes by name
- Process graphical analysis
  - Depth
  - Breadth
  - Interconnectedness
  - Impact

# Frontend Functional Requirements

- Display graphical visualization of data to the user such that:
  - Nodes and edges are clearly distinguishable
  - Graphical analysis (depth, impact, etc.) is intuitively displayed by color and shape
- Provide an interactive interface for the graph
  - Select nodes to display the node's data
  - Pan around the graph to explore the data
  - Display to the user when data is being processed

# Non-functional Requirements

- Maintain data integrity (no modifications)
- Process and visualize data over 32GB in size
- (Obviously) don’t crash


Linux Web Server
Python Server
Web Browser
Graph Panel
Graph Handler
Node Properties Panel
Search Utility Panel
HTTP
Rest Service
Graph Processing
Websocket
Neo4J

# Backend Specifications and Design

- Receive graph csv file via REST
  - Data is then parsed and loaded into neo4j
- Websocket server
  - Handles various requests
  - Requests and responses are in JSON format
- Neo4j
  - DB that holds edges, nodes, and properties

# Frontend Specifications and Design

- Graph Panel
  - Display graphical visualization of data
  - Allow user interaction with the graph
- Search Utility Panel
  - Select data from the server to visualize
  - Search within a graph for specific data or neighborhoods
- Node Properties Panel
  - Display information on a node selected by the user

# Tech Stack

- Python
  - Server language for interfacing with Neo4J, graph file uploading, and UI communication
- React.js
  - JavaScript framework used for writing the user interface
- Vis.js
  - JavaScript graph visualization library that provides an interactive graph, given a dataset
- Neo4J
  - Used for graph storage and handling

# Work Breakdown

- Agile practices
  - Three, two week sprints
  - Trello
- Git
  - Feature branches for new development
  - Pull requests with team code reviews

# Fall 2015 Milestones

- September
  - Gather requirements and project information
  - Establish communication with client
- October
  - Research relevant technologies
  - Begin prototype implementation
- November
  - Complete prototype for first semester

# Current Progress

- First iteration of user interface complete
  - Visualization of a small graph ~(100 nodes, 450 edges)
  - Graph selected by providing data name in the Search Panel
  - Basic user interaction such as node selection, with data displayed in the Node Properties Panel
- FTP data file upload to server

# Individual Contributions

- Backend Team
  - Michael
    - Created automatic file upload service to submit files to Neo4J.
  - Richard
    - Wrote backend handlers for frontend requests.
    - Communication with Neo4j.
- Frontend Team
  - Alberto
  - Andrew
  - Taylor
- Graph Panel, Search Panel, and Node Properties Panel
- Graph Visualization through Vis.js

# Challenges

- Tech stack
  - Everything was new technology to the team
  - Switched from Angular to React
  - Switched from D to Python
  - New techs reflect what Workiva commonly uses, anyway
- Scalability
  - Large data sizes
  - How to improve performance without sacrificing interactivity

# Next Steps

- Gather performance metrics based on graph size
  - Calculated estimated 17% increase in data size from CSV to JSON graph data
  - Improve data transmission for much larger data sizes
- User Experience feedback for UI improvement
  - Search Panel becomes a file selector instead of pure search
- Visual graphical analysis