Project Proposal

**INTRODUCTION**  
The project (<https://github.com/allaccountstaken/fdic_banks_eda>) will provide explanatory data analysis (EDA) of banking industry evolution. It seems that the industry is going through a rapid consolidation as a number of regulated entities declined from 8,500 to roughly 5,000 in the last 20 years.

From a preliminary survey it appears that only 500 banks actually failed during these years and, consequently 3,500 disappeared due to other reasons, such as mergers, charter changes and voluntary liquidation. The main objective of the EDA is to create visual explanations of this consolidation, i.e. historical trends and drivers.

**THE DATA**  
Federal Deposit Insurance Corporation (FDIC) is a United States government agency that examines and supervises the majority of commercial and savings banks (<https://en.wikipedia.org/wiki/Federal_Deposit_Insurance_Corporation>). The Corporation collects and maintains granular datasets describing the industry as a whole as well as individual regulated entities.

Statutory Reports of Condition and Income (Call reports) are provided by the Examination Council via SOUP API: <https://cdr.ffiec.gov/public/ManageFacsimiles.aspx>. General data are available to public via REST API: <https://banks.data.fdic.gov/bankfind-suite/>. Detailed Swagger documentation is available here: <https://banks.data.fdic.gov/docs/swagger.yaml>. Moreover, FDIC web sites contain numerous HTML tables with aggregated information that can be obtained programmatically with Python requests library.

Insights we are aiming to produce are important because credit institutions are critical for community development and economic growth. Moreover, costs of financial restructuring, although are not directly endured by tax payers, are still unproductive and avoidable. Finally, understanding consolidation drivers can help inform direction fintech innovation and future product development.

**EXPERIMENTAL DESIGN**  
The data will be obtained from the official FDIC databases. Historical events will be queried using REST API and statutory reports will be obtained using SOAP connection. It is anticipated that two separated Python clients will be developed. Queries will be executed using report dates and unique entity identifiers, i.e. ID RSSD or FDIC Certificate Number.

It is proposed that EDA will be following the waterfall chart pattern, <https://en.wikipedia.org/wiki/Waterfall_chart>, i.e. quarterly status: new entrants, mergers and spinoffs, and exits. This will require a tabular data format: rows being report dates and bank IDs, columns being event status and selected financial metrics. Expected format of the resulting table is provided below:

| Report Date | Bank identifier (ID RSSD) | Events & Changes | Total Assets ($M) | Other financial metrics… |
| --- | --- | --- | --- | --- |
| 09/01/2019 | 12345XCV | New entrant | 120 |  |
| 01/01/2020 | 12345XCV | Active | 150 |  |
| 04/01/2020 | 12345XCV | Charter change | 250 |  |
| 07/01/2020 | 12345XCV | Liquidation | 50 |  |

**PROJECT MANAGEMENT**

1. Finalize this project plan
2. Connect to SOAP API, get a list of available IDs and dates (starting from the year 2000), get a list of available financial metrics (up to 50), develop client, test, pull quarterly data, store in .csv, expected (400,000 rows, up to 50 columns)
   * 1. Second pair of eyes to do independent client testing.
     2. Data cleaning, drop missing demographic and impute missing financials. Alternative approach, TBD.
3. Research and develop a more efficient storage, i.e. SQL database v Apache parquet file, something else TBD?
4. Connect to Events REST API, get a list of available events, cross check available IDs and date range, develop Python client, test, pull events, up to 10 categories expected, one-hot-encoding transformation, merge with SOAP dataset.
   * 1. Second pair of eyes to do independent client testing.
     2. Demographics data, Name and State, should match SOAP dataset.
5. Additional data from table scraping: list of failed banks with dates, list of costs associated with failed banks.
   * 1. Demographics data, Name and State, should match SOAP and REST dataset.
6. Develop CLI to do manual querying of the dataset, at least for date ranges, bank IDs, names and states, selected financials, such as Total Assets and Net Income.
   * 1. Second pair of eyes to do independent CLI testing.
7. Research and develop visuals, ideally waterfall chart. What library to use? If in Jupyter Notebook decide plotly vs bokeh? Should we aim for user interaction with plots, i.e. dates selection? Should we use tableau?
8. Draft detailed report, update README on GitHub, check consistency of code documentation following PEP guidelines.

**RESULTS**  
*How to display and discuss the results?*

This depends on #7 decision, TBD.

**TESTING**

At least three separate modules need to be tested. Consider testing strategies, unit testing v boundary testing:

1. Python client class to connect to SOAP API, separate logic in .py file, connection may require security token, hide in environment variable, exclude from GitHub, provide use case screenshots, time the queries.
2. Python client class to connect to REST API, separate logic in.py file, provide use case screenshots, time the queries.
3. Python command line interface CLI to query the data file, and or SQL database. Should we provide executable or a .py? TBD.

**OUTCOME**  
*Summarize your plan and explain how your findings could be used by others (if applicable).  
   
Include your team number and all team members in the proposal. That is, one group member submits on behalf of the entire group as a group assignment. The purpose of the proposal submission is for feedback (not for grading).*

* Dima Mikhaylow, [agp7dp@virginia.edu](mailto:agp7dp@virginia.edu)
* [Lauren Bassett](https://app.slack.com/team/U02B0CW6RDZ), [jme9rt@virginia.edu](mailto:jme9rt@virginia.edu)
* [Cullan Bedwell](https://app.slack.com/team/U02B0CWDA59), [qja3fh@virginia.edu](mailto:qja3fh@virginia.edu)
* Case, [ddr3de@virginia.edu](mailto:ddr3de@virginia.edu)