Things to do:

* Figure out layout + requirements
* Develop user stories
* Finalize JIRA layout

General concept: A web application that is capable of managing member/club information for the Creighton Pep Band. Information managed by the web app is as follows:

* Member information (grade, attendance score, netID, music preferences etc.)
* Folder inventory (a collection of all music folders and their contents owned by the Creighton Pep Band, along with their current lending status and corresponding owner)
* Instrument inventory (a collection of all instruments owned by the Creighton Pep Band, along with their current lending status and corresponding owner)
* Song inventory (a collection of all songs owned by the Creighton Pep Band, along with information pertaining to songs currently in circulation for use (in setlist), and tags pertaining to the songs)

User stories:

* As a freshman, alto saxophone player, and a prospective new member of the pep band, I want to be added to the band roster. I would also like to request a new instrument to borrow as well as a folder. By doing this, I can meet the requisites necessary to participate in the band.
  + A member-level user of the app should be able to:
    - Put in a request to join the band, and be authorized to access the site after creating a user account
    - Put in a request for a new instrument and folder based on their respective section
  + An executive-level user of the app should be able to:
    - Add Alice to the members table, along with corresponding data for year, section, netID, and default values for her attendance score (0), and preferences (None)
    - Search for an available, playable alto saxophone from the band music inventory, and assign it to Alice
    - Search for a complete alto saxophone folder and assign it to Alice, as well as recording her payment info and lent date
* As a sophomore, clarinet player, and owner of folder C23, I recently lost my copy of the song, “White and Blue”, and I want to request a new copy of the song so that I can play the song when requires (ie: practices and games)
  + A member-level user of the app should be able to:
    - Send a request to the site owners that his folder, C23, requires the song White and Blue
  + A member-level user of the app should not be able to:
    - Access or send requests for other people’s folders
    - Edit any data on the site
    - Access any executive-level app functions
* As a member and vice president of the pep band, I am in charge of managing attendance and corresponding attendance scores for the members of the band. However, I am not familiar with SQL, and I currently use a google sheet or excel file to keep track of member attendance scores. I want to be able to upload a copy of this file to the app so that I can update member attendance scores.
  + An executive-level user of the app should be able to:
    - Easily upload a spreadsheet to the app to update member attendance scores
    - View, sort, and filter members by attendance score and other relevant attributes (ie: section)
    - Easily search for and modify individual member attendance scores
* As a student of Creighton university, I’ve taken interest in the pep band and I am considering joining. I want to learn more about the pep band, and potentially request to join the pep band so that I can participate in club activities.
  + A general-level user of the app should be able to:
    - View information about the club, such as who we are and what we do, as well as executive contact information
    - Put in a request to join the pep band
* As a member and treasurer of the pep band, I have recently been elected as a new executive for the club. I currently possess a member-level account, but I want to elevate my site privileges, so that I can fulfill my duties as the current treasurer.
  + A member-level user of the app should be able to:
    - Submit a request for a change of privilege levels
  + An administrator-level user of the app should be able to:
    - View and change privilege levels of users

Frontend layout:

* App has multiple functions for different authorization levels, general, member, executive, and administrator
  + Functions are differentiated with a login page, which allows for login using netID or email and password
    - Passwords stored in YAML file, accessed and authenticated with streamlit\_authenticator, refer to [this link](https://blog.streamlit.io/streamlit-authenticator-part-1-adding-an-authentication-component-to-your-app/)
  + Registration page:
    - Page to setup user account with fields for netID, email, and password, checkbox to request executive level account, checkbox for new members
    - For general accounts:
      * If new member checkbox is filled, redirect to page to input grade, section, and if instrument is needed (sends request to executive/admin-level users)
    - For executive accounts:
      * User is not created, request sent to admin-level users to authorize and create account
  + General-level:
    - Home page of the site, containing information about pep band and request to join
    - Page to contact current executives
    - Account setup page (upon creating account, user is granted member-level privileges)
  + Member-level:
    - Page to submit lost music/folder report and, in turn, request a new folder/music
    - Page to view upcoming events/calendar page? (optional, low priority)
      * At least have links to direct to calendar and signup pages
    - Page to view and edit account information (view: member info [without attendance score], email, password, netID, privilege level edit: password, preferences)
      * Button to redirect to privilege level change request page
        + Page has a selectbox with privilege levels (member, executive, and administrator) aside from current privilege level, and a submit button to place a request with administrator-level users
  + Executive-level:
    - Page to view, search for, sort, add, remove, and edit songs by tags, setlist status, practice score, preference score (via member preferences), etc.
    - Page to view, search for, sort, add, remove, and edit members by grade, attendance score, section
    - Page to view, search for, sort, add, remove, and edit folders by section, completeness, lent status, paid status, etc.
    - Page to view, search for, sort, add, remove, and edit instruments by section, ownership, condition, etc.
    - Page to generate setlist based on uploaded signup page as a .csv file (optional, low priority)
      * At least be able to generate specific priority scores based on attendance sheet
    - Page to upload attendance sheets to change member attendance scores
  + Administrator-level:
    - Page to view, search for, sort, add, remove, and edit all app users and their respective info (ie: privileges, etc.)
    - Page to view all changes made to app (optional, low priority)
    - Page to view incoming requests and authorize executive-level account creation

Frontend layout requirements:

* For search pages: use selectboxes to apply filters to dataframes, then use [data\_editor](https://docs.streamlit.io/develop/api-reference/data/st.data_editor) to apply changes and have a submit button to save and rewrite changes
* For upload pages: use file upload widget to get respective file and have data reader for scores
  + Alternatively, figure out if there is streamlit-google suite compatibility (optional, VERY low priority)
* For add pages: Create a page with fields based on the corresponding data table (ie: member, instrument, folder, song, etc.). Allow for submission/addition as long as all fields are filled with proper information
* For request page: Figure out a way to authenticate users? Maybe login necessary for requests so that people don’t mess with other people’s data/requests
  + Alternatively, make requests require manual changes, users are unable to alter data
* Implement changelog to view edits and hold accountability

Backend requirements:

* Create temporary/practice data to use while transitioning real data to SQL database
* Implement functions for dataframe sorting and editing
  + Create corresponding SQL statements for each function
  + Use st.connection and .query for SQL statement calls

JIRA layout:

* Sprint 1 (9/3/24 - 9/17/24):
  + Tasks:
    - Figure out layout + requirements
    - Develop user stories
    - Finalize JIRA layout
  + Backlog:
    - Clarify frontend layout? Things may be missing
* Sprint 2 (9/17/24 - 10/1/24):
  + Tasks:
    - Work on/finalize backend functions
    - Create phase 1 presentation
* Sprint 3 (10/1/24 - 10/15/24):
  + Tasks:
    - Create practice datasets to work with while SQL server is unimplemented
    - Create frontend page structure, add sidebar for page navigation and implement privilege-based access
    - Create request page layout, implement backend functionality
* Sprint 4 (10/15/24 - 10/29/24):
  + Tasks:
    - Create login page, implement backend functionality
    - Confirm functionality of login page with practice accounts at each privilege level
    - Create search page layout, implement backend functionality
    - Create add page layout, implement backend functionality, customize for each search page
* Sprint 5 (10/29/24 - 11/12/24):
  + Tasks:
    - Begin work on/complete SQL server and data
    - Connect SQL server to app, ensure security
* Sprint 6 (11/12/24 - 11/26/24):
  + Tasks:
    - None, buffer sprint for backlogs
* Sprint 7 (11/26/24 - 12/3/24):
  + Tasks:
    - Create phase 2 presentation