A list of deviations from what was proposed in PRJ566

Note:-

Text in red - What we missed
Text in green - What we achieved
Text In Blue – Additional new features

1. **User Management System:**
    1. Front End Development:
        1. Develop frontend UI for new user email verification.
        2. Create User Account:
            1. Design and create UI components for “Create Account” page and popup form.
            2. Implement form validation and submission functionality.
            3. Integrate frontend with backend API for account creation.
            4. Implement redirection to the homepage after account creation.
        3. User Sign-In:
            1. Develop UI components for the “Sign-In” page and popup form.
            2. Implement for validation and submission functionality.
            3. Integrate frontend and backend API for user authentication.
            4. Implement redirection to the homepage after successful sign-in.
        4. Edit Profile Info:
            1. Design UI components for the “Edit Profile” page and popup form.
            2. Implement form validation and submission functionality for profile editing.
            3. Connect frontend to backend API for updating user profile data.
            4. Implement redirection to the profile page after editing.
            5. Store a session cookie to enter the application and be able to restrict the pages
        5. Delete Profile:
            1. Create UI elements for the “Delete Account” page and confirmation dialog.
            2. Implement confirmation logic and handling for account deletion.
            3. Connect frontend to backend API for deleting user accounts.
            4. Handle redirection to the “Edit Profile” page or other relevant pages after deletion confirmation.
        6. View Other User Profile:
            1. Design UI elements for Displaying other user profiles.
            2. Implement UI components for viewing other user profiles.
            3. Integrate frontend with backend API to retrieve and display other user profiles.
            4. Implement redirection to the “View User Profile Page” when viewing other user profiles.
        7. Forgot password.
            1. Design UI components for the “Forgot Password” page and popup form.
            2. Implement form validation and submission functionality.
            3. Verify that the entered email is in the correct format.
            4. Check if the email is associated with an existing account.

5. Display appropriate error messages if necessary.

8. Change Password
    1. Design UI for the Change Password page, where the user can enter their current password, the new password, and confirm it.
    2. Includes clear instruction messages, especially regarding new password requirements

2. Backend Development:
    1. User Authentication and Verification:
        1. Develop backend logic for verifying email for new user accounts.
        2. Implement authentication mechanisms for user account creation and sign-in.
    2. User Account Management:
        1. Create backend endpoints to handle account creation, editing, and deletion.
        2. Implement data validation and storage for user account information.
    3. Profile Information Retrieval
        1. Develop backend endpoints for retrieving other user profile information.
        2. Implement logic to fetch and serve user profile data securely.
3. Database Schema Design:
    1. Design database schema for storing user account information.
    2. Define database tables for user profiles, authentication, and related data.
4. Data Storage Implementation:
    1. Setup database structures for storing user accounts, profile information, and authentication data.
    2. Implement CRUD (Create, Read, Update, Delete) operations for user-related data.
5. Forgot password backend.
    1. Connect the password reset form with the backend API to submit the new password
6. Change Password backed.
    1. Create a change password API endpoint that accepts a POST request with the current password and the new password.

2. Journal Management System:

1. Front End Development:
    1. Develop frontend UI for the ViewJounal Screen.
    2. View Journal:
        1. Design and create UI components for “View Journal” page, including the floating “+” icon to create and add a new Journal and Edit and Delete Journal options.
        2. Integrate frontend with backend API to get data and showcase it on Journal View Page.
        3. Implement redirection to Journal View Page after viewing one Journal.
        4. Implement navigation to Add Journal Entry Form Page when clicking on “+” icon.
        5. Design and integrate a hamburger menu. The menu provide quick access to different sections such as edit, delete, and change status.
        6. Implement a rating system that allows users to rate journal entries with stars.
        7. Implement a like button that allows users to like journal entries and keep track of the number of likes.

3. Add New Journal:
    1. Develop UI components for the “Add-Journal” page which is a Add Journal Entry form page.
    2. Implement text input and attachments in the form, implement form validation, and Discard and Add Entry functionality.
    3. Integrate frontend and backend API for posting newly added journal into the database.
    4. Implement redirection to the Journal View Page after successfully adding a Journal to showcase newly added Journal.
    5. Add Google Maps integration for location-based journal entries.
    6. Allow users to search for and select locations using the map interface.
    7. Add functionality for changing the privacy status of the journal entry.
    8. Add support for adding videos using Dropbox integration.
4. Edit Journal Entry:
    1. Develop UI components for the “Edit-Journal” option which is a Edit Journal Entry form page.
    2. Implement text input and attachments in the form, implement form validation, and Discard and Edit Entry functionality.
    3. Integrate frontend and backend API for updating journal into the database.
    4. Implement redirection to the Journal View Page after successfully editing a Journal to showcase updated Journal.
5. Delete Journal Entry:
    1. Create UI elements for the “Delete Entry” option and confirmation dialog.
    2. Implement confirmation logic and handling for Journal deletion.
    3. Connect frontend to backend API for deleting user journals.
    4. Handle redirection to the “Journal View” page to showcase the entry being deleted, if selected yes, and when selected no, should get rid of the confirmation pop-up, and redirect back to Journal View Page.

2. Backend Development:
    1. Journal Management:
        1. Create backend endpoints to handle Journal creation, editing, and deletion.
        2. Implement Journal Entry data validation and storage for user’s Journal Entry information such as entry text, date, and attachments.
        3. Ensure that the ratings and likes data is retrieved from and stored in the backend using the appropriate API endpoints.
    2. Journal Information Retrieval
        1. Develop backend endpoints for retrieving user’s Journal Information such as date, title, attachments.
        2. Implement logic to fetch and serve user journal data securely.
        3. Authenticate your app users with Dropbox to obtain a user access token.
        4. Store the videos in the Dropbox cloud to later be consumed by an API
3. Database Schema Design:
    1. Design database schema for storing user’s Journal Entry information.
    2. Define database tables for user Journal Entry’s title, date, entry text and attachments and related data, if there are any such as User Id, etc.
4. Data Storage Implementation:
    1. Setup database structures for storing user’s journal information, and authentication data.

2. Implement CRUD (Create, Read, Update, Delete) operations for user's Journal Entry data.

3. User Timeline System:

1. Back End Development:
    1. User Authentication and Registration:
        1. Develop backend endpoints for creating and deleting travel journal entries in the Timeline.
    2. Database Schema Design:
        1. Design database schema for storing user's Timeline information.
        2. Define database tables for user Timeline and the access to the Journal Table
    3. Data Storage Implementation:
        1. Implement Create and delete operations for Journal Entry into the Timeline data.
        2. Optimize database queries and operations for performance.
        3. Design a scalable database architecture to handle user growth.
        4. Conduct performance testing to identify and address bottlenecks.
2. Frontend Process for UserTimeline Feature:
    1. Develop frontend UI for the UserTimeline Screen.
    2. View Timeline:
        1. Design and create UI components for "View Timeline" page, including the floating "+" icon to add a new Journal to Timeline
        2. Integrate frontend with backend API to get data and post new journal to timeline.
        3. Implement navigation to Journal Entries Page when clicking on "+" icon.
        4. Implement redirection to Timeline with the new Journal entry in the timeline.
    3. Add New Journal Entry to Timeline:
        1. Develop UI components for the "Add-JournalToTimeline" page.
        2. Implement redirection to Journal Entry to Timeline View Page.
        3. Using API POST add the selected journal entry and then redirect again to Timeline View Page
    4. Delete Journal Entry:
        1. Create UI elements for the "Delete Journal in Timeline" option and confirmation dialog.
        2. Implement confirmation logic and handling for Journal Entry deletion.
        3. Connect frontend to backend API for deleting Journal entry in Timeline.
        4. Handle redirection to the "Timeline View" page to showcase the entry being deleted, if selected yes, and when selected no, should get rid of the confirmation pop-up, and redirect back to Timeline View Page.

4. Forum Feature Development

1. Front End Development
    1. Forum Main Page UI
        1. Develop UI for the main page of the forum.
    2. Thread Creation UI
        1. Design and implement UI for creating new message threads.
    3. Thread Response UI
        1. Develop UI components for responding to threads.
    4. Forum Navigation
        1. Implement navigation within the forum (e.g., search).

        2. Add Google Maps integration for location-based journal entries.
        3. Allow users to search for and select locations using the map interface.
    2. Back End Development
        1. Thread Management Logic
            1. Develop backend logic for creating and managing threads.
        2. Response Management Logic
            1. Implement logic for posting and managing responses.
    3. User Participation Features
        1. User Profile Integration
            1. Display User Profiles in Forum
            2. Implement feature to display user profiles in forum interactions.
        2. Interaction Features
            1. Develop functionality for users to like and react to forum posts.
            2. User is able to reply each forum.
    4. Database and Infrastructures
        1. Database Schema for Forum Data
            1. Design Database for Threads and Responses
            2. Create database schema for storing threads and responses.
    5. Data Storage and Retrieval
        1. Setup CRUD Operations for Forum Data
            1. Implement Create, Read, Update, Delete operations for forum data.
        2. Ensure Data Security and Privacy
            1. Implement measures for data security and user privacy.

5. Profile Page

1. UI Design:
    1. Create a user interface for the Profile Page that displays the user's personal information.
    2. It includes sections to display data such as name, email address, profile image, and other relevant details.
2. Data visualization:
    1. to get and display user information from the backend.

6. Settings Page

1. UI Design:
    1. Design the Settings Page that includes sections such as About Us, Change Password, Feedback, and Edit Profile.
2. Navigation:
    1. Implement navigation to allow users to access each section of the settings page.
3. Change of password:
    1. Provides a form for users to change their password.
    2. Includes fields for current password, new password, and confirmation of new password.
4. Feedback:
    1. Offers a section where users can give feedback about the application.
    2. Implement a form for users to submit comments.
5. About Us:
    1. about the application, company, or team.
6. Profile editing:
    1. to the profile editing section within the settings page for easy access.

7. Github

1. Branches:

    1. Each developer created a branch to work independently on their part of the project.
    2. This facilitated parallel development and allowed changes to be pulled and pushed in a professional manner.
2. Continuous Integration (CI):
    1. A continuous integration (CI) system was implemented to automate the deployment of local changes to production.
    2. Production Implementation:
3. For the frontend
    1. Vercel was used to implement and manage the deployment.
4. For the backend
    1. Cyclic was used for production deployment. These environments enable fast and efficient deployment, ensuring that changes are tested and automatically deployed to production.