InReach Dev Setup Notes

First Task: Start with trying to get InReach installed locally following the instructions below.

- ***Secondary project note*** *(after setup locally & more familiar with the code): a few team members can create a virtual machine environment (for scalability; to help streamline the initial onboarding process for future ENG volunteers/contractors)*

These are draft setup notes and may contain errors. These notes were written based on running the app locally in a macOS environment. If you are on windows the syntax may be different. We do welcome updates to these notes so if you find something is not correct, please do suggest edits in this google doc. Once these notes have been reviewed, they will make their way into the github repo and will fall under source control.

Getting Started

Prerequisites

pnpm (node package manager)

This project uses [`pnpm`](https://pnpm.io/) to manage packages. To install, run the command:

```bash
npm -g install pnpm
```

or follow the instructions on [pnpm's installation page](https://pnpm.io/installation).

As of the writing of these instructions, node version v20 or higher is expected. Ensure your system has this version installed and is in use.

Docker

Docker (& docker compose) are used for local databases. Instructions to install Docker [can be found here](https://docs.docker.com/get-docker/)

Install docker ui as well

Project specific setup

1. Create a project dir
2. Open vs code (or code editor)
3. Initialize vscode for git if not already done

4. Https clone repo (InReach) - https://github.com/weareinreach/InReach.git
5. Select above project dir as destination
6. Choose to open the project
7. When vscode fully opens/clones the project, select to open the workspace file
8. Set up the .env file (there are several .env files, under the app project level(InReach root), the database project level (Centralized database), and the storybook project level (Shared UI). Start by placing the .env in the app project level. It might also be needed at the database level and Shared UI level.
    a. Copy the .env.example file within the InReach (root) dir and create a new .env file, placing it within the InReach (root) dir
    b. Open the new .env file and generate your secrets for NEXTAUTH_SECRET=and SESSION_SECRET=. Ensure each has it's own value
        i. from the terminal and enter 'openssl rand -base64 32'
        ii. Copy the resulting value to the NEXTAUTH_SECRET line
        iii. from the terminal and enter 'openssl rand -base64 32' a second time
        iv. Copy the resulting value to the SESSION_SECRET line
    c. Add this EDGE_CONFIG value to the .env file
        i. EDGE_CONFIG=https://edge-config.vercel.com/ecfg_1sqfggbdhoelhs9pm6mlhv951pfu?token=bab31db6-b629-4a7d-829d-a1afd51a9dcd
    d. Add the google places api
        i. GOOGLE_PLACES_API_KEY=AIzaSyC5iYumi3Do2AMD4FqUudu2WX_X4swDN3M
    e. Add the FEATURE_FLAG, COGNITO_USER_POOL_ID, CRON_KEY, NEXTRUNTIME variables
        i. COGNITO_USER_POOL_ID=us-east-1_06XOmcvrs
        ii. CRON_KEY=wcJnXEy/MWBZ06nKDIvh/3mpTWu2+amQ4ndfvJkyK9w=
        iii. FEATURE_FLAG_CONFIG=https://edge-config.vercel.com/ecfg_ucmezh24kfagkbjlfr1kjokmdwhy?token=cf9acc8d-2141-4332-a8d7-cd43bdebfa20
        iv. NEXT_RUNTIME=nodejs
        v. AWS_REGION=us-east-1
        vi. AWS_DEFAULT_REGION=us-east-1
    f. Add the GH_DATASTORE_PAT value (this is used for crowdin data migration)
        i. Generate one from github
    g. Make a copy of this edited .env file and save it to the app, db, and ui paths
        i. app path: InReach/apps/app directory (from the side panel this is the 'InReach App (@weareinreach/app)' section
        ii. db path: InReach/packages/db directory (from the side panel this is the 'Centralized database (@weareinreach/db)' section
        iii. ui path: InReach/packages/ui directory (from the side panel this is the 'Shared UI (@weareinreach/ui)' section
9. From the project root ensure node is installed
    a. npm -g install pnpm
10. From the project root install dependencies
    a. pnpm install

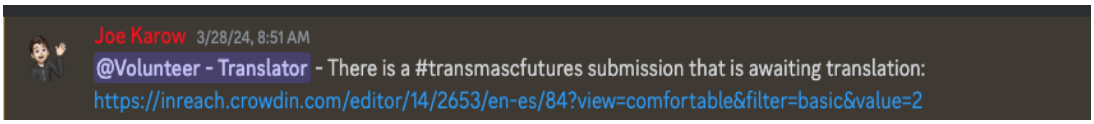
11. From the project root, start docker
    a. pnpm docker:up
12. Add support for dotenv-cli
    a. cd to the apps/app dir
    b. run, pnpm install dotenv-cli --save-dev
13. Add at least one location to the locales.mjs file
    a. Open the locals.mjs file from within InReach/packages/db
    b. Ensure the 'localeList' array has the value 'en'
    c. Save the file
14. From the app directory, start the app project
    a. cd to InReach/apps/app
    b. pnpm dev
15. A bunch of scripts run, look for the Next.js value -which should show the Next.js version of 14.2.16 or something similar. Under that is the value for 'Local', this is the url path for where the app can be found. Default is 'http://localhost:3000'

Integration Details

Crowdin:

- Crowdin is a platform used to manage and automate language translations. Inreach integrates with it using github actions triggered by a workflow_dispatch event_trigger.
- The Crowdin GitHub Action synchronizes files between GitHub and the Crowdin platform.
- It runs on a schedule and/or when changes occur in specific branches like `dev` or `main`.
- It uploads source files to the Crowdin project (based on the branch name) for localization.
- When translations are updated on Crowdin, the action pulls the changes back into the corresponding branch (e.g., `dev` or `main`) on GitHub, ensuring up-to-date translations.
- The developer workflow should be:
    - Create a branch `locally` for the text update
    - Make the update to the specific english language file
    - Merge the changes from local branch into the `dev` branch
    - Wait for the Crowdin action to occur for the dev branch, then from the editor panel within Crowdin, choose a lange and find the text that changed

- Copy the resulting URL to the Discord #translation channel, making a request for an updated translation. Something like this:



CI/CD Details:

How to do a release?

- What we don't know is what was/is the specific process Joe used to deploy new code onto the production server
    - Start by looking through the code for deployment scripts (might be a migration script via Vercel; might already be automated through GitHub)
    - This goes together with the below (i.e., Git workflow)
- As of Jan 2025 the release to production process is as follows:
    - Review any open PRs and merge them as needed into the dev branch
    - Verify the merge and deployment process ran successfully by going to appstaging.inreach.org (if one has access to vercel, they can also go to the vercel dashboard nad verify the deployment is successful)
        - Verify the needed db migration scripts ran correctly
        - Verify ui/us changes are as expected
    - Verify there are no github action errors. If there are any analyze and fix or determine if it is safe to proceed
    - Create a PR to merge changes from dev into main
        - Title the PR with something like 'release <today's date>
        - In the body of the PR list the changes - can be simple like the change ticket titles
    - The engineering lead will review the PR and merge if there are no other changes needed

Git workflow/process:

Looks like there are several github actions that handle pushing and pulling files from crowdin
Integrating with vercel - vercel looks to have deployment scripts configured. Included in these scripts is to run dataMigrations as part of the deployment.

*There are many open automated PRs created from various tasks/actions. How/when/what is the process for merging and closing these? These should be reviewed before being merged.
https://github.com/weareinreach/InReach/pulls?q=is%3Apr+is%3Aopen+release

Database:

How to get seed data locally?
How to see data in Production/Preview?
How to trigger a migration?

Install yout UI tool of choice for easier viewing of the data.
pgadmin is one such tool - https://www.pgadmin.org/download/

List of Databases:

- postgres - default administrative connection database
- inreach-dev
- inreach-prod
- analytics
- rdsadmin

The env file has a reference to SNAPLET. Presumably this is where anonymized data from the DB exists (or existed). One of the setup steps is to get a signup link from the team lead and connect to this platform so the local DB can be populated with anonymized data. However, SNAPLET as a service no longer exists and the entire tool has been made open source.

TODO: define a process/method to seed the local DB
TODO: define/determine what the purpose of the various databases listed above and how they interact with one another

Database servers on AWS

InReach Dev (server)

this a server with several databases - don't know what url uses this data
Has 5 databases:

- analytics,
- Inreach-dev
- Inreach-prod
- postgres,
- rdsadmin

InReach Prod (server)

this a server with several databases
Has 5 databases: looks to be used by appstaging and production

- analytics,
- Inreach-dev: used by appstaging
- Inreach-prod: used by production
- postgres:
- rdsadmin:

I've tried to take a snapshot of this data (pgadmin bkup) and restore it to my local "inreach" DB, but the restore fails every time for a variety of issues.

I've had moderate success with exporting individual tables (vi pgadmin), then importing them into a temp DB. I then export from the temp DB to my local 'inreach" DB. Not the best process but might be good enough to get the needed data for testing.

Migration scripts

There are several migration scripts listed in the package.json file in the Centralized database repo. When run locally, they run but some steps result in errors.

- db:migrate - runs without error. The seems to setup the core database structure
- db:dataMigrate - runs with errors. This adds and edits data such as geo data, permissions, links, etc. Some of these migrations fail.

The best results, thus far for getting a DB running locally are:
pnpm db:reset
pnpm db:deploy
Restore from the local_db_bkup file
pnpm db:dataMigrate (run as needed to update data changes. Ie. if data has changed from the last production snapshot)

Working notes:
With pgAdmin4 ui tool - connect to postgres db and open query tool

```
None
SELECT pg_terminate_backend(pid)
FROM pg_stat_activity
WHERE datname = 'inreach';
```

With pgAdmin4 ui tool - connect to postgres db and open query tool
Delete ‘inreach’ db if it exists

```
None
DROP DATABASE IF EXISTS inreach;
```

From pgadmin4 ui
Create a new db called ‘inreach’ with owner ‘user’

Note: the db running on docker is postgresql. It is important when trying to connect to the Docker db that no other postgresql service is running locally, else pgAdmin4 will fail to connect with a ‘user does not exist’ error, or something similar.

Data Import for local DB

There’s been some mild success with exporting and importing tables individually.
Import in this order:

- Created a snapshot of the production database manually and imported it locally - local_db_bkup (this should be stored for safekeeping so it can be shared if needed)
-

User roles and permissions:

The following tables control user access, roles, and permissions:
user, userPermissions, Permissions

- user - list of user accounts
- permissions - specific tasks a user can perform
- userPermissions - links users to permissions

Storybook setup

Follow steps above to get the InReach primary app running
Cd to the root of the storybook path: ..InReach/packages/ui
Ensure all packages are updated, run ‘pnpm install’
Copy the .env file from setting up Inreach to the root of the storybook path
Start story book: ‘pnpm dev’
If you run into issues with ‘with env’ you may need to Add support for dotenv-cli

a. From the storybook root run, ‘pnpm install dotenv-cli --save-dev’

Draft notes for getting the DB running locally

TransMascFutures App/web

Summary

TransMascFutures is …..

There are several branches in github but the ones of most concern are:

- dev - base local development off this branch, i.e. create new branches from dev
- main - used in production, don't merge directly without first branching from dev

The basic workflow is:

- git checkout dev
- git pull
- git checkout -b <new branch name>
- git commit
- git push origin <new branch name>

Then on github

- Create a PR
- Merge into dev
- After review and on whatever "schedule" exist, merge dev into main
- Deployment scripts run, which includes any db schema or data migrations, new code is pushed to production

Local Installation

***Important** - TransMasc and InReach use the same port numbers when running locally. To reduce confusion the suggestion is to run only one project at a time.

1. Create a project directory
2. Clone this repo: https://github.com/weareinreach/TransMascFutures
3. Ensure the pnpm package manager is installed then
    - From the root directory run 'pnpm install'
4. Ensure docker is installed (used for the database)
5. Create the .env file and add the following items (copy the .env.example file and update the NEXTAUTH_SECRET value)
    - # Environment

- NODE_ENV=development

- # Prisma
- # Database connection URL for Prisma
- POSTGRES_PRISMA_URL=postgres://user:password@localhost:5432/postgres

- # Database connection URL without connection pooling
- POSTGRES_URL_NON_POOLING=postgres://user:password@localhost:5432/postgres

- # Next Auth
- # You can generate the secret via 'openssl rand -base64 32' on the command line
- # More info: https://next-auth.js.org/configuration/options#secret

- NEXTAUTH_URL=http://localhost:3000
- NEXTAUTH_SECRET= # generate with `openssl rand -base64 32`, this can also be the same secret that's been set for the InReach project

6. Setup the database
   On local, you will be using the docker container, specifically the 'postgres' db
   - pnpm db:up - this will start the docker containers for the db setup
   - pnpm db:migrate:apply - this will create the tables and seed the db
7. Verify the DB configuration
   - From the docker UI, under the 'docker' container there should be 2 database views: adminer-1, db-1

     To use Adminer (a web-based database management tool):

     - Open your browser and go to http://localhost:8080.
     - Enter the following details:
       - **System**: PostgreSQL
       - **Server**: `db` (if inside Docker network) or `localhost` (from the host machine)
       - **Username**: `user`
       - **Password**: `password`
       - **Database**: Leave blank to list all databases or use `postgres`.
8. Start the TMF application
   - From root type 'pnpm dev'
9. To use storybook for component viewing, development, changes etc
   - From root type 'pnpm dev:ui'

Database

Local DB (on docker) settings

- User: user
- Password: password
- DB: db (postgres)
- URL: postgres://user:password@localhost:5432/postgres

Production DB settings

- User: default
- Password: DsTtpFb16XRJ
- DB: verceldb
- URL: ep-dry-resonance-088928-pooler.us-east-1.postgres.vercel-storage.com

Adding user stories & Translations to the website

- Users can fill out surveys on the web to submit their story.
- The details from the survey are saved to the StoryToCategory, Story, StorySubmission, Pronouns, PronounsToStory tables.
- A request is sent to Crowdin to have the survey answers translated
    - Via an API call in the story.submit function the response1 and response2 values are sent to the “new-submissions.json file in the TransMasc Crowdin project, https://inreach.crowdin.com/u/projects/14
    - There is also a Discord webhook configured for the TransMasc project. When the new-submissions file is updated, a notification is triggered to appear in the #translations channel in the Inreach Discord
- Once translated and a manual review of the data by ENG to ensure the data is not “trolling”, engineering will
    - Create a db migration script - use a previous script as a template - when run, this updates the Story table by setting the published value to true as well as adding the spanish translations
    - Save the script to the data-migrations folder
    - Test the script locally - run ‘pnpm db:dataMigrate’ - ensure the response fields for EN and ES are completed and the published flag is true in the DB
    - Create a PR then merge this change following github workflow protocols
    - Once the PR is merged into main, the DB updates will be deployed to production via github actions
    - Verify new story appears in production

Creating the dataMigration script

- Open the ‘!YYYY_MM_DD_new-job-template.ts’ file under /prisma/data-migrations
- Follow the steps documented a the top of the file
    - // Copy the file and make the name of the new file today's date plus the storysubmission id (i.e. 2023-06-27_cm2m6ae8l0000aik98z887ke6)
    - // Complete the values under the section for 'Define the job metadate here'
    - // for Production the data is in the 'vercel' db
    - // for local the data is in the 'postgres' db
    - // The *ES values will be in the new-submission file on CrowdIn - search by the storyId
    - // test the migration locally by running 'pnpm run db:dataMigrate' from the command line
    - // (note, you must use local data, if it works, update the data to be that from production)
    - // save the file, push changes to github, then create a PR
    - // create a migration file for each story to be added or updated
    -