REYKJAVIK UNIVERSITY

T-404-LOKA

Einkenni.is - Operation Manual

Baldur Olsen
balduro19@ru.is

Kári Steinn Hlífarsson
kari19@ru.is

Magnús Friðrik Helgason
magnush19@ru.is

supervised by
Stefán Ólafsson

May 12, 2023

Contents

1 Introduction

In this manual, we will discuss how to run, update, and maintain the system einkenni.is. This manual will feature an in-depth description of how to run the code behind the software on your local machine as well as the necessary steps required to make changes.

2 Git-Source control

Git is a distributed version control system that allows multiple team members to collaborate on a project, track changes to files, and manage the history of the project. It provides a robust way to maintain the integrity of the source code and manage the development process efficiently. This chapter will guide you through the basics of using Git for source control management in the context of the Einkenni.is project.

2.1 Setting up Git

Before you can start using Git, you need to have it installed on your machine. Visit the official Git website at https://git-scm.com/downloads to download and install the appropriate version for your operating system. Once installed, you can check if Git has been installed correctly by running the following command in your terminal or command prompt:

git –version

2.2 Cloning the Repository

To begin working with the Einkenni.is project, you need to clone the remote repository to your local machine. To do this, open your terminal or command prompt and navigate to the directory where you want to store the project. Next, run the following command, replacing the URL with the actual repository URL:

git clone <repository-url>

This will create a local copy of the project on your machine, allowing you to make changes to the files and track their history.

3 Starting the Backend (API)

To start the API the first step is to navigate to the root directory of the project called Einkenni.is. From there you can do this from your terminal/command prompt window or open the root directory in your preferred text editor such as visual studio code. Below is an example of the file structure opened with visual studio code on macOS.

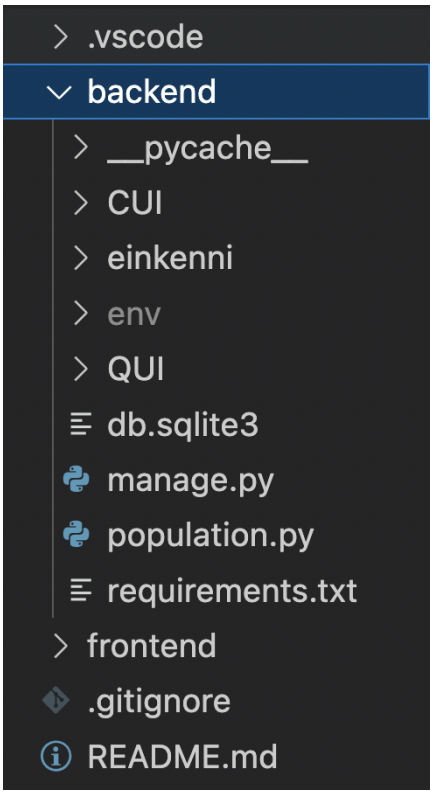

Figure 1: File setup in Visual Studio Code

For the first time you open the directory shown in figure 1, the env folder will not be present. Navigate to the back-end directory in your terminal. You must install the virtualenv library by running the following in your terminal window.

pip install virtualenv

In order to run the API you will need to set up your own virtual environment. To create the environment run one of the following commands.

virtualenv env
or
python -m virtualenv env

If done successfully you should see an env folder pop up in the back-end directory similar to figure 1. If this is not the case please visit the following website for more information: https://docs.python.org/3/library/venv.htr
Whilst in the back-end directory after having created the env folder run the following command in your terminal window to initialize the environment.

windows: env/scripts/activate
macOS/Linux: source env/bin/activate

After running your terminal should display the following (*env*) next to your username.

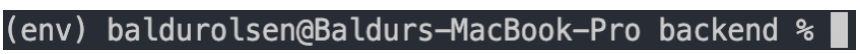

Figure 2: Terminal with initialized env

Now the env is all setup and ready to go, you'll need to install the requirements for the software to work. In your backend directory (Figure 1) there should be a requirements.txt file. If the file is not present please re-visit the git repository and acquire it. For this next process you will need to install those requirements by running the following command in your terminal:

pip install -r requirements.txt

If done successfully you can now start the API. In the terminal run the following command:

python manage.py runserver

The API is now up and running on your localhost and the terminal should display the following:

```
System check identified 1 issue (0 silenced).
May 11, 2023 - 18:19:22
Django version 4.1.6, using settings 'einkenni.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Figure 3: Server running at localhost

Congratulations you have successfully started the API!

4 Starting the Frontend

In this section, we will guide you through the process of starting the frontend for both the questionnaire and consultation interfaces. The frontend is developed using the React framework and can be easily started by following the instructions provided below.

To start either the questionnaire or the consultation interface, every single step beside one is the same. The first step is to navigate to the root directory of the project called Einkenni.is. From there, locate the "frontend" folder. When starting the questionnaire interface, open a terminal or command prompt window and navigate to the "frontend/questionnaire" directory. When starting the consultation interface, open a terminal or command prompt window and navigate to the "frontend/consultation" directory.

If this is your first time running the frontend, you will need to install the required dependencies. Run the following command to install all necessary packages.

npm install

After the installation is complete, start the development server by running the following command.

npm start

The development server should now be running, and the terminal will display the address where the application is accessible (usually, it is http://localhost:3000/). Open your preferred web browser and navigate to the provided address to view and interact with the questionnaire interface.

```
Compiled successfully!

You can now view questionnaire in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.38:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Figure 4: questionnaire interface running at localhost

Figure 5: consultation interface running at localhost

5 API File System

From the backend directory seen in figure 1. Multiple folders and some files reside there. The db.sqlite3 file is an SQLite file that contains the database of the project. Manage.py is the main file in any Django system and is used to manage various aspects of the Django project, run $python manage.py help$ for more. population.py is a file specific to this project and was used to populate the database from .CSV files that were already acquired. More on how to use DRF: https://www.django-rest-framework.org/.

5.1 einkenni-Files

The ´einkenni´ folder is the main Django project. All files are generated by Django at the start of the project. The important files seen in the following figure, are the settings.py and URLs. The settings.py file in Django is a Python module that contains all the settings for a Django project. these settings include: The database configuration, Installed applications, Middleware classes, Timezone settings, Static files configuration, Template configuration, Email configuration, Security settings, and much more.

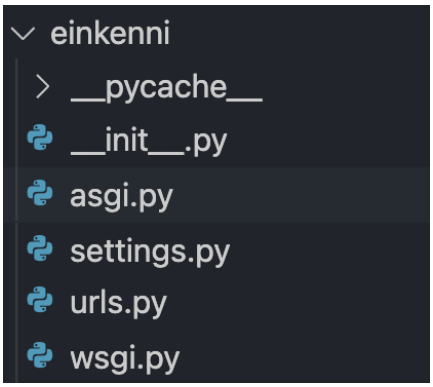

Figure 6: Einkenni folder

the urls.py file in the project application serves as the main URL configuration for the entire project. It acts as the central hub for managing the routing of URLs to their corresponding views.

5.2 QUI-Files

QUI files are the questionnaire part of the API. The classifiers are for the machine learning model. The admin.py file is used to register database models to the admin portal. The DiagnosticModel.py contains the class responsible for getting data from the model itself. The generate_questions.py file is responsible for creating and maintaining the question list.

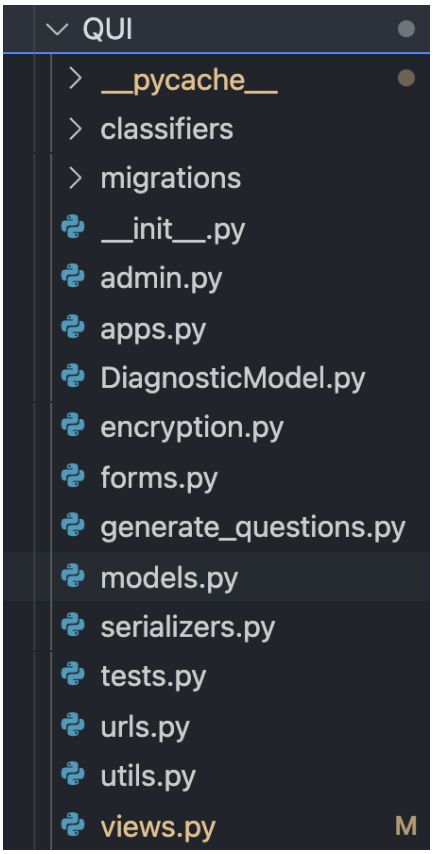

Figure 7: QUI folder

Other important files seen in figure 7 are those for interacting with the database, models.py, serializers.py and forms.py are all data managing tools used in Django, see more here: https://www.djangoproject.com/.

5.3 CUI-Files

The file system for the consultation API is very similar to that of the questionnaire interface but does not contain the machine learning model itself as all relevant information can be cast between the two.

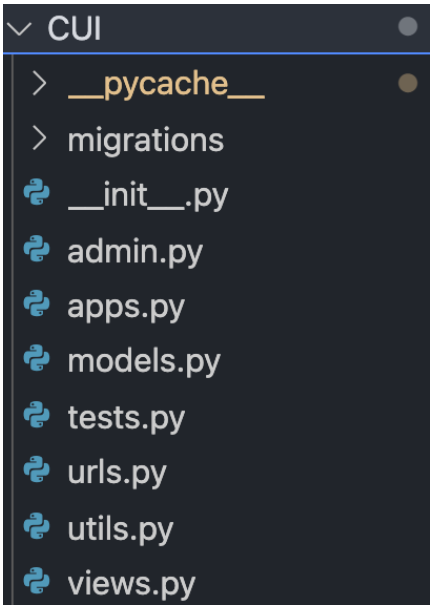

Figure 8: CUI folder

6 React File system

From the front-end directory, two separate folders are available: The consultation folder and the questionnaire folder. Each is a React application.

6.0.1 Consultation folder

When opening the consultation folder, you are presented with the following directory:

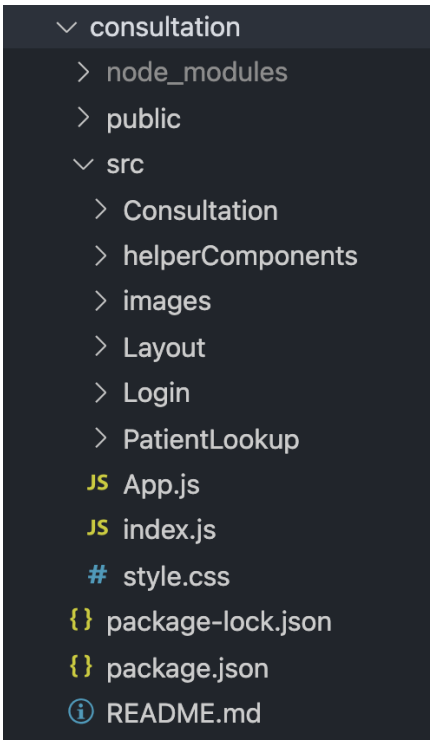

Figure 9: Consultation folder

The README.md file shows the basics of React, such as how to run a development server, how to build the application for production, etc). The *node_modules* folder will not be there initially but will appear after running the ´npm install´ in the command line. The public folder includes the index.html file, which is needed to start the application, but typically needs not to be changed at all. The ´src´ folder includes the majority of the relevant information, such as the App.js file, which includes the routing for the different URL patterns for the application. The various folders all include either a component for rendering in the DOM, or a helper component/code to support the files rendered with the routing protocols used from App.js.

Following are the routing paths from App.js, in the consultation/src directory:

```
function App() {
  return (
    <BrowserRouter>
      <Layout>
        <Routes>
          <Route exact path='/' element={<Login />} />
          <Route path='/patient-lookup' element={<PatientLookup />} />
          <Route path='/consultation' element={<Consultation/>} />
        </Routes>
      </Layout>
    </BrowserRouter>
  );
}
```

Figure 10: Consultation routes

6.0.2 Questionnaire folder

When opening the questionnaire folder, you are presented with the following directory:

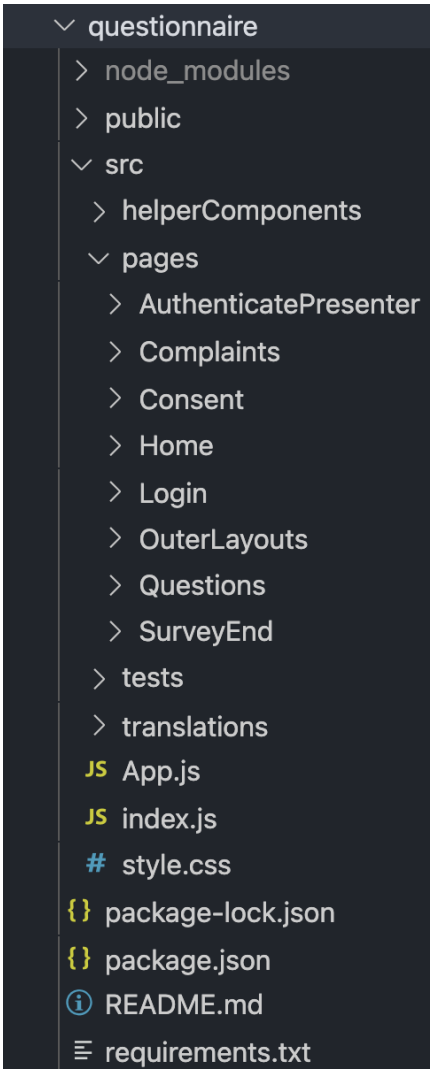

Figure 11: Questionnaire folder

The same layout is used for the questionnaire as in the consultation folder, except for the translations folder (including two JSON files with contents in English and Icelandic), and the additional ´pages´ folder, which distinguishes the components that are used to render contents in the page from

the helper components and code. Each folder within the ´src´ folder (excluding the ´helperComponents´) is paired with a route from the routing paths in App.js.

Following are the routing paths from App.js, in the questionnaire/src directory:

```
return (
  <BrowserRouter>
  <AnimatePresence>
    <ScrollToTopOnMount />
    <Routes>
      <Route path='/' element={<Home languageState={languageState}/>} />
      <Route path='/consent' element={<Consent languageState={languageState}/>}/>
      <Route path='/authenticate-presenter' element={<AuthenticatePresenter languageState={languageState}/>}/>
      <Route path='/ssn-login' element={<Login languageState={languageState}/>}/>
      <Route path='/symptoms' element={<MainComplaints languageState={languageState}/>}/>
      <Route path='symptoms-prioritization' element={<ComplaintsPrioritization languageState={languageState}/>}/>
      <Route path='/questions' element={<Questions languageState={languageState}/>}/>
      <Route path='/survey-end' element={<SurveyEnd languageState={languageState}/>}/>
    </Routes>
  </AnimatePresence>
  </BrowserRouter>
);
```

Figure 12: Questionnaire routes

7 Heroku

Heroku is a cloud-based Platform as a Service (PaaS) that simplifies deploying, managing, and scaling applications. It supports a wide range of programming languages and frameworks, making it a popular choice for deploying web applications, including the questionnaire interface. The questionnaire interface is already deployed on Heroku and is accessible at the following URL: https://einkenniqui.herokuapp.com.