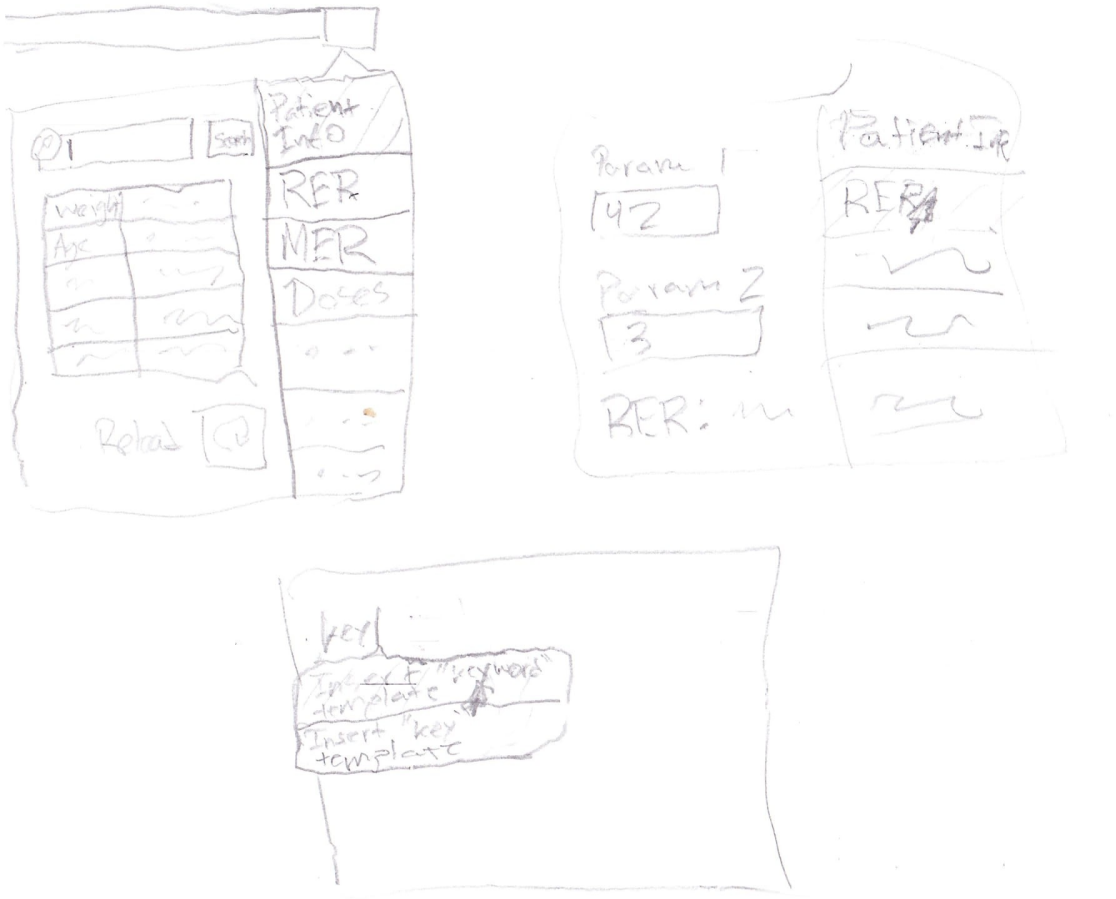
Criterion D: Product design—Overall structure

The extension will have a menu with the tools my client requested. The following mockup, which I shared with my client, demonstrates the basic structure of the extension. Upon clicking the extension's icon, a dropdown of tools appears. When hovering over an option with the mouse, the tool is shown on the left. The first tool gives information about the patient that the user is currently looking at, as well as allowing the attachments in that patient's record to be searched.

Calculators, like the RER calculator shown, automatically update their result when the parameters are changed.



The text editor at the bottom of the mockup suggests templates to be autofilled based on keywords being typed. These can be clicked on to be inserted.

Overall structure of software solution

The product will have the following components:

WebExtension compatible with Firefox and Chrome

The WebExtension standard is largely supported by both Firefox and Chrome. Extensions can be made compatible with minimal changes. (Source: MDN)

Criterion B goes into more depth about the details and scope of the extension.

Mock server for the EHR site

Due to obvious security and confidentiality concerns, I cannot give access to my Client's EHR site to graders. This creates a need for a server that mocks the behavior of my Client's EHR site.

The server will be made with Flask. It is impossible to correctly mock everything about a proprietary service like my client's EHR site; the goal will be to roughly mimic the behavior of features of the site that I am extending, such that the extension will work both on the mock server as well as the real website.

Internal structure

List of resources and techniques

The resources used are organized into several categories:

Software

Resources	Details
Firefox	
Chrom[ium]	
web-ext	
webpack	
Git	
Node	
NPM	**Several packages, including the Mozilla web-ext tool will be installed locally**
Python 3	**For the mock server; will install additional libraries like Flask locally in a venv**
Vim	**Optional**

Platform-specific package manager	**Optional, for installing the listed packages**

Software libraries

Resources	Details
pdf.js	**https://github.com/mozilla/pdf.js**
moment.js	**https://github.com/moment/moment**
Bootstrap	**https://github.com/twbs/bootstrap**
Flask (python)	**https://github.com/pallets/flask**

Resources provided by client

Resources	Details
Editor template files	**Format has not yet been agreed upon with client**
Mathematical formulas	**For calculating things like Resting Energy Requirement (RER)**
Logs of HTTP requests to the EMR site	**Will be saved as HAR files**

Tasks/Techniques

Technique	Details
Arrays and iterators	**Data, such as attachment files, will be stored in an array or processed as an generator/iterator stream.**
String handling	**Information from the EHR may need to be parsed using regular expressions and document text will be compared with a query string.**
Object-oriented design	**Use of classes for abstracting data about patients**
Asynchronous programming	**The extension will handle HTTP requests asynchronously**
Use of CSS	**Extension tools in the menu will be stylized with CSS, probably using bootstrap**
Analysis of HTTP requests sent by the browser	**HAR files will be saved with the Firefox dev tools and analyzed using Python to learn how the EHR site responds to different requests**
Use of Flask to create a simple website (mock server) based on specifications (behavior of the EHR site)	**Largely, this will just be a static website, but we will need to mock certain dynamic features of the site (see "Analysis of HTTP requests sent by the browser")**
Use of version control (git)	**Files will be tracked with git and frequent commits will be made.**

Test plan

Test item	Test data	Part of system	Expected	Actual	Comments	Ref in

		tested	outcome	outcome		product
Cover page testing – REQUIRED ELEMENT FOR CRITERION G						
Web pages load from the cover page in three different locations	**File naming to ensure home page is called Index.htm**	**Links on cover page are relative**	**Loads as required from 3 different locations**			
Product testing						
Mock server can be run without the need to install dependencies	**Mock server bundle**	**Creation of a self-contained PyInstaller exectutable**	**The mock server executable runs without errors**			
Relevant features of the EHR site work on the mock server	**Mock server**	**Correctness (to a reasonable degree) of the mock server**	**The mock site appears to behave correctly; un-mocked features clearly indicate that they are not mocked**			
Extension bundle can be installed easily in the browser	**Extension bundle**	**Creation of extension bundles**	**Extension bundle installs successfully**			
Information	**Extension**	**Extension**	**Information is**			

about patients extracted		**behavior**	**correct and when the EHR is missing data, the extension indicates that it is missing.**			
Hover selection	**Extension**	**Extension behavior**	**Hovering on dropdown menu reveals tools in extension popup**			
Live calculations	**Extension**	**Extension behavior**	**Calculators automatically update results when parameters are changed**			
Document search	**Extension**	**Extension behavior**	**Documents that match the search pattern should be shown. Searching should be case insensitive.**			
Textareas offer suggestions	**Extension**	**Interface of template code with website**	**When entering keywords into textareas on the page, template suggestions appear below**			

Agreement of client

I confirm that the requirement specification meets my needs and the designs above are appropriate for the creation of the product.

- Camille Fischer (Client)

Signature: Camille Fischer