An Interactive End-To-End Machine Learning Platform

Implementation Report

April 18, 2021

Requirements Specification	**Murat Kurnaz**
Design	**Mustafa Enes Batur**
Implementation	**Ömer Erdinç Yağmurlu**
QA / Testing	**Tarek Gaddour**
Final	**Atalay Donat**

Contents

1 Tools

1.1 VS Code

Code Editor

1.2 PlantUML

Charting software

1.3 MikTEX

Latex compiler and package manager

1.4 npm

node package manager

1.5 create-react-app

an intuitive command line scaffolding application easing the development of react applications

TBD, ADD MORE STUFF HERE, PYTHON ETC

2 Challenges

2.1 Client

2.1.1 Testing Clients

The front-end is written in React and is composed of presentational components (components), stateful components (containers) and hooks. In separating presentational and stateful components from one another we wanted to ease testing. Although testing presentational components took place without a problem using Jest.js snapshots, render tests and some simple consistency tests, stateful containers were harder to test, since they required extensive mocking of React's hook and lifecycle events.

2.1.2 Sensor Data Collection

Browser Inconsistencies

from https://github.com/PSE-TECO-2020-TEAM1/client/issues/5#issuecomment-817308653

There are two main APIs on sensor access in browsers right now, namely the Sensors API, which is incorporated into the web standard and supported by 71.03% of all users worldwide.

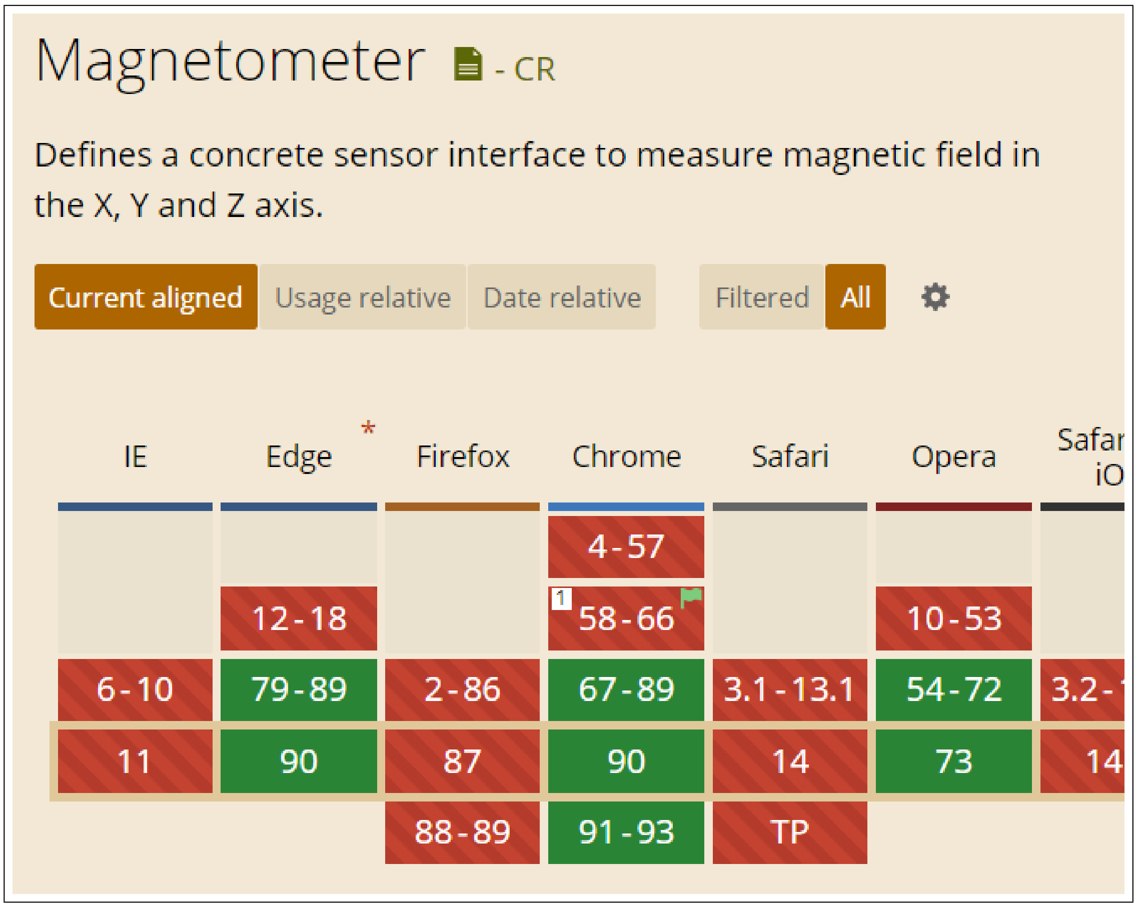
On the other hand there is the legacy DeviceMotionEvent API, which was an experimental technology designed before the aforementioned Sensors API was drafted. It is supported by 94.97% of users worldwide (albeit with handicaps).

In this application, we are using the new standard Sensors API, which is only supported by Chrome and Chromium based browsers like Edge, Brave etc. for now. Apple refuses to implement the new API citing privacy concerns, there is no information on why Firefox doesn't implement it. Since on Apple platforms all browsers from all vendors use the Safari WebView, this new API doesn't work at all on Apple devices.

The DeviceMotionEvent API is unfortunately not suitable for use at all. All three different major browsers (Chrome, Safari and Firefox) have a different understanding of what coordinates they return and have no documentation of which units they return the data in.

We've tried to use a polyfill with the branch sensors-polyfill, but were getting totally different results with different browsers (and with firefox totally broken results. Because of this reason we've decided not to support Apple users at all.

Magnetometer Originally we wanted to support Magnetometer sensor too, since it was (supposedly) supported by the browsers we were targeting (on caniuse.com).

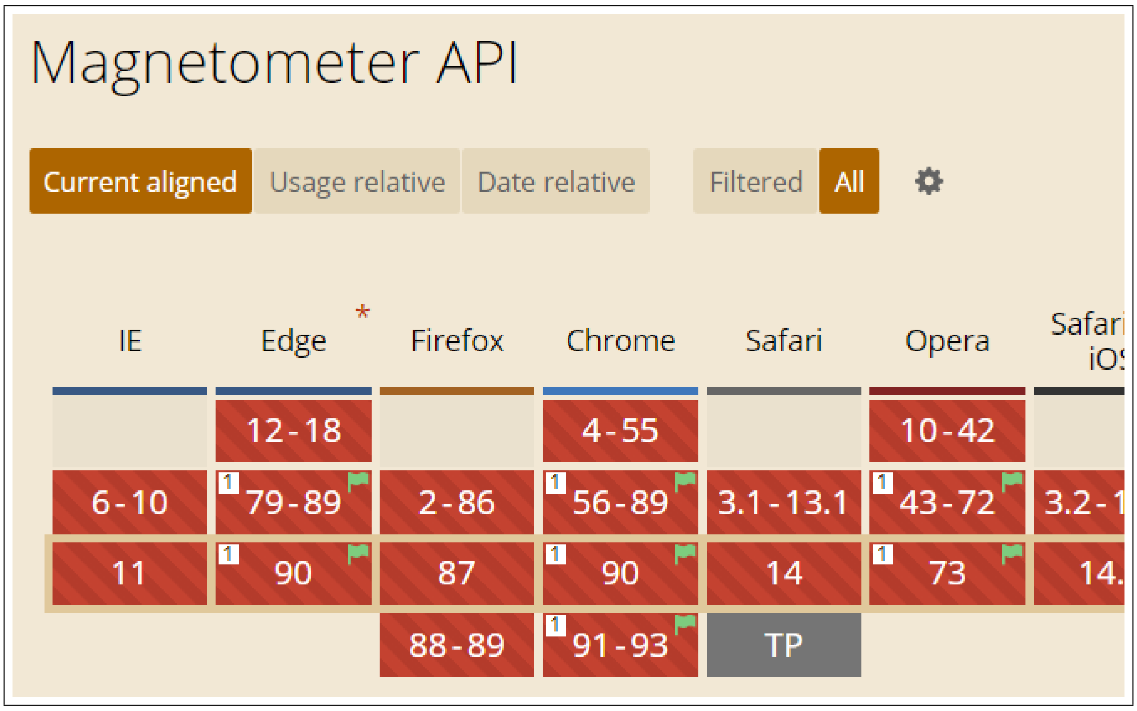

Magnetometer - CR

Defines a concrete sensor interface to measure magnetic field in the X, Y and Z axis.

IE	Edge *	Firefox	Chrome	Safari	Opera	Safar iO
			4-57			
	12-18		1 58-66		10-53	
6-10	79-89	2-86	67-89	3.1-13.1	54-72	3.2-
11	90	87	90	14	73	14
		88-89	91-93	TP		

Figure 1: caniuse.com - Magnetometer

After implementing the sensor, however, we've noticed how no matter what we've tried, we weren't getting any data, and after more investigations we've discovered that the 'Magnetometer Sensor' and the 'Magnetometer Sensor API' are two different entities separate from each other, and dropped support for magnetometers.


Magnetometer API

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari iO
	12-18		4-55		10-42	
6-10	[1] 79-89	2-86	[1] 56-89	3.1-13.1	[1] 43-72	3.2-1
11	[1] 90	87	[1] 90	14	[1] 73	14.
		88-89	[1] 91-93	TP		

Figure 2: caniuse.com - Magnetometer API

2.2 Workspace Management

Making a Consistent Server and Covering Failure Cases As the workspace management handles storage of user data, it acts as a bridge between the client and the model management. Thus the validity of these data is a very crucial part of the work. Although the client prevents most of the invalid requests, such invalid requests could be handcrafted and sent to the server or the client itself could possibly generate an invalid data by an error so each request needed to be validated. Finding the error cases and handling them generated a lot of work, because finding the not so obvious error cases needed a general analysis of the action caused by the request on the server and possibly on the model management.

Performance-Space Trade-Off Decisions TBD

2.3 Model Management

2.4 Auth

3 Statistics

3.1 client

Lines of code	TBD
Test coverage	TBD
Number of commits	RBD

3.2 workspace-management

3.3 model-management

3.4 auth-management

3.5 Total

4 Changes from Design

4.1 Workspace-Management

4.1.1 Label

Label isn't saved as an embedded document anymore. To achieve the same functionality label now keeps the workspaceId of the workspace it belongs to and the number of samples it is the label of.

4.1.2 Data Point

ObjectId is now omitted as it is not used and causes disarray in the corresponding responses.

4.1.3 Sensor Data Point

ObjectId is omitted here as well on the same grounds. In addition name of the sensor is also added to schema to simplify communication with the client.

4.1.4 Sample

Sample does not embed the label data anymore, instead it saves the id of the label. Implementation of setTimeFrames method is also delegated to the workspaceController.

4.1.5 Workspace

TBD

4.2 client

4.2.1 Single Codebase

In the design document, we'd envisioned two different codebases for the different edge (mobile) and management (desktop) clients. During development, however, it has become obvious that a single codebase with client side routing and bundle separation using tree shaking was more suitable for our application. We are bundling both applications in a single router, and there is no clear separation between each client. During development, special care was given to reduce cross dependencies between both parts to a minimum in order to reduce the bundle size for edge devices.

4.2.2 /lib folder

Originally, there were only two auxiliary classes in the whole client (MobileAPI and DesktopAPI), while everything else was a React component. During development, we made use of custom react hooks (/lib/hooks) to refactor common stateful logic into reusable parts. Apart from hooks, sensor data collection needed it's own abstraction over the clunky Web API implementation, which we've placed in /lib/sensors.

4.2.3 API endpoints

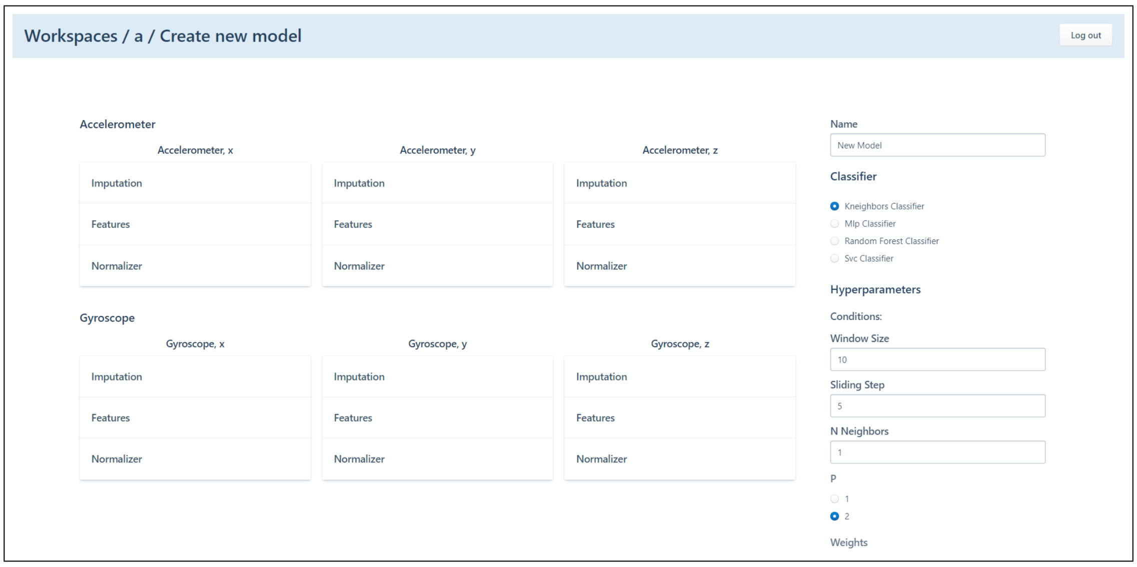
In order to accomodate changes in the backend, both the Mobile- and DesktopAPI have undergone major changes in the interfaces they implement.

4.2.4 Component Library

In the design document, we'd specified Material-UI as the component library that we were going to use. During development we've found it too clunky, heavy and hard to develop with and replaced it with the Evergreen Component Library from segment.io. This component library also comes with it's own opinionated 'CSS-in-JS' styling library, through which we were able to style the application with a rapid pace.

4.2.5 ModelOptions Component

This component faced the most changes during implementation. We'd decided to distribute some model creation options to the sensor-components themselves instead of using the same set of parameters for each component. This had it's implications on the client and it had to be updated accordingly.



Workspaces / a / Create new model

Log out

Accelerometer

Accelerometer, x	Accelerometer, y	Accelerometer, z
Imputation	Imputation	Imputation
Features	Features	Features
Normalizer	Normalizer	Normalizer

Gyroscope

Gyroscope, x	Gyroscope, y	Gyroscope, z
Imputation	Imputation	Imputation
Features	Features	Features
Normalizer	Normalizer	Normalizer

Name: New Model

Classifier

[✓ Kneighbors Classifier]
Mlp Classifier
Random Forest Classifier
Svc Classifier

Hyperparameters

Conditions:

Window Size: 10

Sliding Step: 5

N Neighbors: 1

P: 1 [✓ 2]

Weights

Figure 3: New ModelOptions

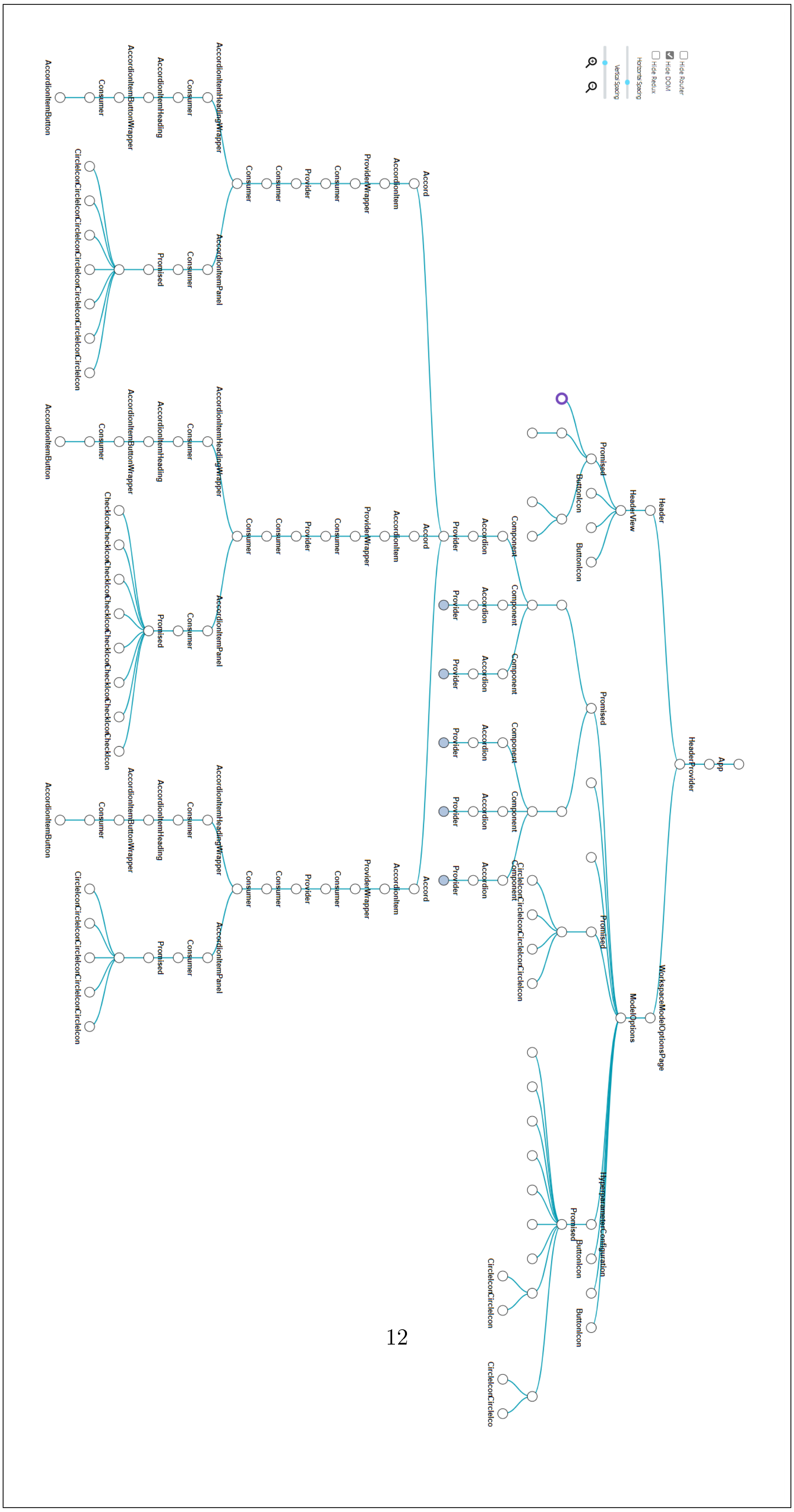


Figure 4: New ModelOptions Component Graph

4.3 model

4.4 auth

5 Functional Requirements Coverage

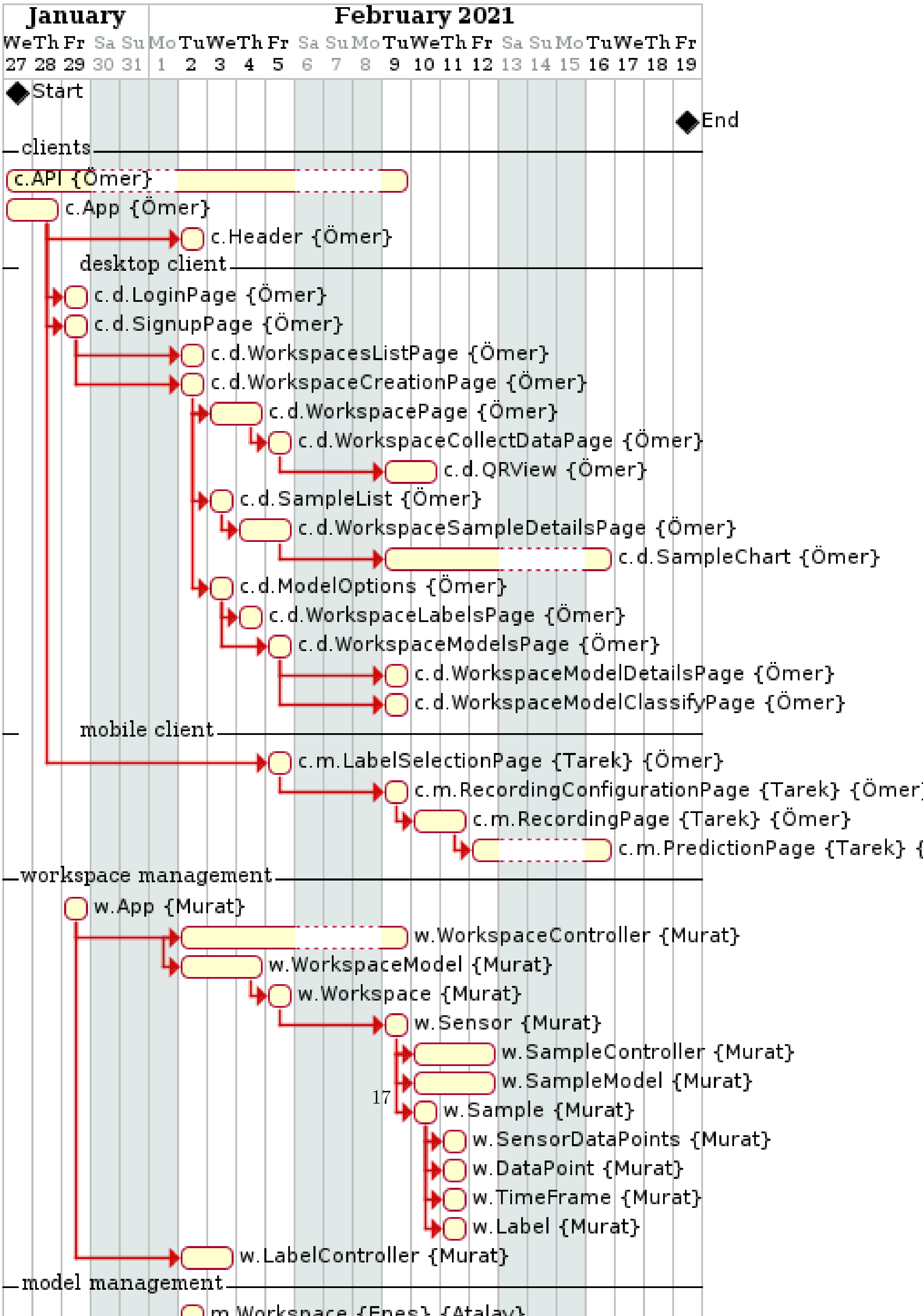
5.1 Mandatory Requirements

Number	Requirement Name	Implemented?	Notes
NUM	Here comes the name	YESNOPARTIALLY	

5.2 Optional Requirements

Number	Requirement Name	Implemented?	Notes
NUM	Here comes the name	YESNOPARTIALLY	

6 Planned Schedule



7 Actual Schedule

TBD: CAN EVERYONE UPDATE THE GANNT CHART IN THE PLAN FOLDER WITH WHAT THEY'VE DONE

8 Unit Tests

TBD