Exam in KWS2100 Geographical Information Systems for the web

This is a practical exam which will evaluate you ability to deliver usable map based web applications using OpenLayers. You will expect to create a working application using the functions that have been taught in class and deploy this on GitHub Pages or Heroku. In order to get a top grade you have to demonstrate data sources, functionality and techniques beyond what was thought in class.

Formal requirements:

1. Work in teams of 2 or 3. All team members must have committed substantial code to the exam repository, or you must describe in the README-file how your work process involved pair programming processes where everyone contributed even though they didn't commit equally
2. Deploy a functional application on Heroku or GitHub pages using GitHub Actions. Your grade will be primarily determined by the functionality demonstrated by your application. If some functionality isn't easy to discover, you should describe this in the README-file to make sure the examiner finds all the functionality you want to be included in the determination of the grade
3. You must create a map that displays several pieces of geographical information on a map. You should demonstrate data sources beyond what has been demonstrated in the lecture and assignment for full credit.

The delivery

- [ ] You must create a GitHub repository using the exam invitation link: https://classroom.github.com/a/XamnFLGl
- [ ] You must upload a ZIP-file with the repository (*excluding* `.gitignore` *d files*) to WiseFlow.
- [ ] You must have a README.md file with a link to your deployed website
- [ ] Your README should contain a description of your project and what you want to be reflected in the grading

NOTE: The GitHub repository will be locked for new changes (pushes) at the time of the deadline for the exam.

Scoring expectation

E: If you deploy a React application that displays a map with OpenLayers on GitHub pages, you will pass. If you fail to deploy correctly, but your code indicates that you could have achieved a C, you will also pass

D: Your map must contain some vector data and some interaction

C: The interactions, data sources or styling is beyond what was expected for the assignment. You need to display data sources that was not used in the lectures or assignment with at least two types of geometries (point, linestring, polygon)

B: You are able to create a full features map application:

- You can display both polygon and point geometries from at least 4 data sources
- Your code is structured to make it easy to find the code corresponding to each data source and in order to make it easy to add more features
- You can click on a feature to display an Overlay with properties of the features
- You display an Overview map. This has not been covered by in the lectures, but is well is a well documented feature in OpenLayers.

Pick **one** of the following:

- [ ] Display moving data on a map using a GraphQL data source. You can use Entur's GraphQL vehicle data (not protobuf) or find a GraphQL websocket dataset on your own
- [ ] Display a Clustered vector data source with OpenLayers. The style of a cluster of features should reflect the size of the cluster. Display a single-feature cluster with a separate style using icons that vary based on a property of the feature. *NOTE: The usefulness and design of your styles will affect your grade*
- [ ] Deploy your own GIS API to a hosted service and display a huge dataset from a PostGIS database to a map, limited to the extent displayed to the user. You can use Hono on Heroku or pick your own backend technology for the server (you get free credits for Heroku with GitHub Student Pack). You must check in the code for the server to the repository
- [ ] Combine several Tile Layers by using Layer opacity and let the user choose and combine layers. At least one of the background layers should be a vector tile layer. The background layers should be in more than one geographic projection
- [ ] Create a fully styled vector tiled background map using Mapzen. You can use the OpenLayers Mapzen example at https://openlayers.org/en/latest/examples/osm-vector-tiles.html as inspiration. You must style at least 5 different types of features (object collection or kind) and you must incorporate texts into the styling. For double

points, incorporate pointer interaction, e.g. display bus routes when hovering over them

- [ ] Let the user draw **and modify** features with at least two types of geometries, including circles with a radius in meters. Store the features in localStorage or with Heroku so that when the user refreshes, the features remain on the map. The user should be able to modify the geometry of the feature and properties that should be reflected in the style of the feature
- [ ] Interaction between data in a sidebar and the view on the map: Show a sidebar with a list of a type of feature displayed on the map, filtered to the visible features. When clicking on a feature on the sidebar, the map should zoom to feature. Since there's only one feature visible, the sidebar can be hidden - but it should be easy to zoom back to the previous view. To "go back", store in sessionStorage the view center and zoom before zooming to a feature.

You can vary the details of the requirements as long as you don't make oversimplifications.

A: You must implement a feature rich application with a backend.

- You must implement a backend with Heroku (or your own choice of backend technology) as described under B.
- You must include a dataset with tens of thousands features that is served from the backend as a Vector Tile Layer.
- You must implement a clustered style as described in B
- You must implement either one of the following described under B: a) Vector Tile background map, b) drawing and modifying points and circles, including saving the drawing to the backend, c) GraphQL or similar real-time datasource, d) synchronizing the sidebar with visible features on the map
- Your application should "tell a story" of what use it would have in the real work and the functionality should make sense with that story. It's okay to have functionality that's not really related to the "story" of your application.

For an A, the aesthetic qualities of your application will be part of the evaluation

Note: Data sources

You can use any data source from geonorge.no, kart.dsb.no or any other source you come across. If the data source has features outside Norway make sure the initial view displays the feature so the examiner doesn't need to hunt for them. List where you found your data sources in your README-file.

What about plagiarism?

Don't worry about being accused of plagiarism (unless you consciously try to deceive the examiner)

All software contains reused components and code that's inspired from stack overflow, AI, lectures or other people's code. I expect similarities between submissions. Using code from the lectures is not cheating, but extremely similar code to the lecture code gives partial credit. Plagiarism is to CONSCIOUSLY TRY to give the examiner the impression that you have created something that you have copied. If you **don't** think you are plagiarizing code, you are almost certainly right.