04/24/2023

Lara Notes:

* Send email on the deadlines and dates for the presentation etc
* Weekly meetings for now => Friday on-site
* Send a mattermost channel link to invite the clients
* Project plan should be presented - small analysis of functional and nonfunctional requirements for the application, create a project timeline for friday
* Credentials for our AWS account on friday

Answers to our questions:

* Frameworks we should use: whichever language we are comfortable with
  + (suggestions: AWS console, AWS blogs for the structural design of the application)
  + hosted on AWS
  + automation
  + CI/CD pipeline: application and infrastructure code (python?)
  + CDK AWS
  + Hybrid with Java and Python
* Low-resource language requirements: Turkish - data set access, our options are mostly limited to what Amazon translate has
* Frontend - up to us (half of the screen is translation the other half is blog) or copy the link of the blog page, the real data will be received from the URL, depending on the language choose 1 language and translate it (or if one part of the paragraph is highlighted, it should be translated)
* When do we get access to the dataset? - email them about this
* When can we get the digital training and the reading list?
* Travel compensation?

Pablo’s Notes:

Suggested to use more agile things to structure our plan

Their learning objective: They want us to familiarize ourselves with the platform and the AI tools. They don’t want their objective to interfere with our TU Delft requirements.

### **Frequency w/ Team**

Meetings weekly with team on Fridays.

We need to add them on Mattermost once we get access.

### **Meeting Friday**

Use that time to check on progress, get feedback, do itterations.

Create Functional and Non-Functional requirements.

Project Timeline

Get feedback on above

Finalize plan and be on track.

AWS account will be shared with us soon (by Friday)

### **Frameworks?**

We can use whatever language we’re comfortable with. Open source design system tool. AWS Console. AWS blog.

AWS Infrastructure. Hosted on AWS, using best practices; deployed and reused artifacts. Create a CICD pipeline, not only application code but infrastructure code aswell. CDK is the infrastructure they tend to use (AWS specific) but we can use a different third party (Teraform)

Possible framework for the UI

### **Low Resource language requirements**

Existing Turkish Data Structure, this is a preference. But we can pick another one ,needs to be supported by Amazon Translate.

### **What would the UI look like**

Up to us how the users interact with it. The most important thing is that we have a infrastructure in place for continuous integration and development.

Cloudscape design framework/system.

Article displayed in one half of the screen and translation on another half.

Copy/paste url block, choose language (1 now but should be scaleable).

if you highlight a paragraph, it highlights the other side. If you edit on one side it edits on the other side.

### **Practical Questions**

Training session sent later today

Dataset will get sent depending on LRR

Office space in the Hague if accompanied by AWS staff. Full remote work is possible.

04/28 - Lara

* Cloud practitioner tutorials are the most significant tutorials. Solid foundation of AWS and a good understanding of which learning path would align better for our career objectives. (cloud practitioner level knowledge at the end of week 8 is expected)
  + we do not get a certification since it takes some time to prepare for it, it is not a hard requirement
  + understand the value of the cloud and the different services, how to approach and navigate the platform
  + certificate would take quite a long time
* Specializations around major technologies (certificates)
* Is the translation one way?
  + English to Turkish, not necessarily bidirectional
  + ability to expand to different languages in the future
* No acceptance testing
* No adding comments or notes to the page
* Nice to have an authentication feature to extend the application - up to us
* compatibility requirements across devices?
  + framework that is used to render could help using it on a mobile device - not really a good way (should or won’t), depends if your frontend framework is adaptable to mobile interface, hide the original part and have the translated version (swipe to see the original vice versa)
  + web interface is really important
  + image should stay in the same place that it was

- are we going to be storing any data, do we need a database?

* + - when you translate one blog post, keep that as a database, store its translation, should work as caching
    - stores the latest blogpost and the translations
    - cost optimization! URL of the images should be kept but not the images themselves
* agile, microservice architecture is used, every component can be a microservice, 4 iterations + 1 iteration (0), write the user stories (what the end user will see), then we create a backlog, then split user stories to tasks (functional, performance and security testing)
* iterations aren’t easy to split by user stories but one interaction for translation and one for sagemaker and then for improvements. We have mainly one use case, but a lot of subtasks. task based iterations instead of user stories.
* mix between waterfall, agile and Vee method.
* AWS general coding method: 2 pizza teams, build the teams with 5-8 people (developer, product owner, product manager, testers)
* How can we make the data an integral part of our design process?
* Data → label data → train mode → evaluate model → final model
* Start working backwards from the customer, from the end product
* Use the architecture to go into the data science domain, what is good enough? do a translation and show it to Turkish people for them to rate it?
* you need to come up with a mechanism where we can list these requirements and get a mechanism to provide feedback. (agile, not heavy weight, not a lot of documentation)
* mechanism where we can share feedback/opinions
* maintainability objective?
* project should be a foundation, so handover documents and usability are important parts of the project
* agile with explicit documentation set, make your code much more readable with lots of comments, use external sources to measure your codes readability
* infrastructure should be a part of the repo
* AWS labs, meant to be used in production
* AWS SageMaker samples

What to change in the report in the requirements section?

Functional requirements:

* Change the amazon logo
* Identifying technical jargon, it shouldn’t be translated
* you can add a functionality to Amazon translate to not translate certain words, since we need to train on those words
* in AWS comprehend, you can identify the specific jargon and keep it the same
* post-processing could also work but not recommended
* move the technical jargon up to should have
* application could allow for the user to search for blogs by title → might be more difficult than it sounds, if you don’t have access to all blog posts it will be hard to go about searching, either could have or probably won’t have
* add search functionality for the blogs you have metadata on
* no editing of the blogs
* highlighted part should be mirrored on the translation
* user could export HTML? could print and render it → could have
* should we also translate the comments? won’t have, only the blog that needs to be output, authors are could have
* what do we do with code? avoid translation, must have
* anything that is a hyperlink → should not be translated
* title should be translated
* users are a could have

Non-functional requirements:

* won’t have mobile device
* in the mobile version of the website can be rendered differently, original and the translated
* more clarification on the devices → say desktop or even just browsers
* technology part covers the essential things, it looks good - they might be could have since they want to offer flexibility, there are other AWS services that could be used instead - mostly academically, for AWS the technology is flexible, for AWS these services can change
* Scalability is a must have, change it from should have
* Consolidate everything belonging to technology on one, more generic sentences like “AWS services will be used”
* must have Amazon translate and SageMaker fine-tuned model
* user guide → won’t have, could be just short instructions on the read me
* monitoring part should be in the should have part (ties in with the enterprise)
* what should we monitor and log? → if we are using lambda monitor the compiler executions, invocation, duration, error comments, success rate, connect to cloud watch
* free vs paid users

—

**Pablo Notes:**  
Pre Meeting:

Cloud practitioner is the base certification. Solid foundation of AWS to understand learning paths.

Need to have Cloud partitioner level knowledge by the end of the 10 weeks.

- value of the cloud

- different services

- approach and navigate platform

- we can persue the certification alone after the project if we want

There are different learning paths after cloud partitioner (solutions architect -> assosciate -> )

(DevOps: systems operations -> developer assosciate -> )

(Specializations: security, networking ML, data analytics.)

During Meeting:

Doesn't need to be bidirectional.

User stories:

-person that want the blog translated

Assign user stories to tasks

- coding

- testing

- Documenting

Task based iterations are preferred over User based iterations

Translate -> SageMaker -> etc

2 pizza teams: 2 5-8 people teams. Developer, Data Scientist, tester, product manager, product owner

--

Data science part shouldn;t be seen as an isolated part but rather an integral part of the design process.

How do we start?

- There are different stakeholders with different perspectives. Lets start with the end product and and adress final product. Work backward from end application and talk about user experience. These discussions should provide a basis of implementation. Think of a lightweight mechanism where we can list these requirements.

We need a mechanism where feedback and opinions are possible. We will need to have discussions about certain features to understand why we make certain decisions.

We need to have a mechanism to ensure maintainability.

#Project should be reusable.#

- Important part of the project

- Ensure that things are properly document

- Make the code understandable and commented

- tools to measure code-readability.

#What is done?#

- Code compiles

- on public website

- two translation models

- discussion about approach used to train models

- preliminary models based on translation

- feedback by human reviewers

- infrastructure code so that it is reusable1

- code is readable/understandable following best practices

- High level guidance on how to use the code (readme to help)

- Something similar to AWS labs (high quality and explainable)

What would you like to see if you were taking over this person's code?

---

Tool to export documentation (Like JavaDocs)

CHANGE AWS LOGO

---

Requirements Discussion:

FR

Must have: Skip monopace, skip hyperlink, Include Title

Could have: 'search functionality' (probably a wont have but it is fine in could have)

\* maybe have a repository with last month's worth of blogs

\* metadata storage in dynamoDB

\* GraphQL to communicate with Dynamo

Could/wont have - The user can 'export' the translation. Include the authors' Translation. BlogPost MetaData. Functionality (hyperlink, share) if Amazon Translate does it we should do it. Other wise we can do it

Wont have - Comments wont be translated, Editing original post shouldn't be commented

\_\_\_

NFR:

Define and clarify Devices (tablets, mobile apps, etc.)

It should use AWS Services but it doesn't need to use those specific services (change must have - Techology is flexible)

Data. If Users, they should be allowed to delete their data.

Wont have mobile support

Scalability (1) should be a must have.

Bring down first 2 of technology. Have a more generic overview on AWS services (not forced to use Amplify, etc).

Enterprise setting application.

Monitoring moved to a should have:

- how many users on platform

If using lambda, monitor concurrent executionsInvocations, durations, error count, success rate, fraud pu

Connect to Cloudwatch for monitoring.

---

THings to be aware of for AWS:

best practices session

Intoroduction to AWS