IDEA-FAST

Identifying Digital Endpoints to Assess FAtigue, Sleep and acTivities in daily living in Neurodegenerative disorders and Immune-mediated inflammatory diseases.

Grant Agreement No. 853981

WP4 – WP Device Specific Data Analytics and Performance Assessment

D4.4: Requirements specification data analytics software package

Lead contributor	Teemu Ahmaniemi (VTT), Meenakshi Chatterjee (Janssen)
Other contributors	Emmi Antikainen (VTT), Diogo Branco (FC.ID), Luan Chen (IMT), Francesca Cormack (CCL), Julian Fierrez (UAM), Alejandro Peña (UAM), Anita Honka (VTT), Dan Jackson (UNEW), Juha Kortelainen (VTT), Xujun Ma (IMT), Haneen Njoum (SARD), Rana Rehman (UNEW), Valentina Ticcinelli (UCB), Clémence Pinaud (Lixoft), Chengliang Dai (ICL), Nguyen Truong (ICL), Kai Sun (ICL)
Reviewers	Stefan Avey (Janssen, WP7), Jerome Kalifa (LIXOFT, WP7), David Verweij (UNEW, WP3), David Wenn (IXS, WP1), Kai Sun (ICL, WP5), Ioannis Pandis (Janssen, WP1)

Due date	30 Nov 2021
Delivery date	31 Dec 2021
Deliverable type	R
Dissemination level	PU

Document History

Version	Date	Description
V0.2	18 Nov 2021	Draft for internal review (WP3, WP4, WP5, WP7)
V0.3	26 Nov 2021	Distribution to Steering Committee
V1.0	31 Dec 2021	Final Version

Table of Contents

1 Abstract

This document specifies the preliminary requirements for the data analysis software package in the IDEA-FAST project. The package was agreed to run in the analytical environment which is a part of the Data Management Platform (DMP) developed in the project. Therefore, the document is written for: 1) the developers of the analytical environment to understand the specific needs of the analysis process; and 2) researchers and analysts to realise the data characteristics and analysis methods developed in the project so far.

The DMP has been developed to store and manage the data collected throughout the project. The analytical environment (AE) will allow users to analyse the project datasets without downloading and keeping a copy of the dataset on their local computers, which further prevents data breaches. The AE provides a web-based UI for users to access remote computing resources. Users can access the AE via web browsers, using their email address and password to securely login. Scientific applications such as Jupyter, Matlab, Tensorboard, RStudio will be available via the AE while the primary programming language is Python. Access to Github repositories which are hosting code to perform the analysis is also supported.

The main aim of the analysis is to predict fatigue and sleep disturbances. The analysis methods presented in this document are based on the analysis pipeline described in detail in deliverables D4.1 and D4.3. They focus on device specific data processing methods divided in to 4 Concepts of Interest (COI): Activity, Physiology, Sleep and Social/Cognitive. Devices of the Activity, Sleep and Physiology COI share similar characteristics: they are collecting data continuously with a sampling rate of 25-250Hz. The features of interest, step count, movement magnitude, heart rate and heart rate variability, are calculated using basic signal processing methods available in Python libraries such as *Scipy* and *Numpy* and the Matlab Signal processing toolbox. With cognitive and social COIs, the data contains responses to cognitive tests conducted twice a day on a tablet, or mobile phone usage logs. The amount of data in these cases is rather low and usually compressed to daily aggregates for further analysis.

Actual prediction of fatigue and sleep disturbances is based on association- and multivariate analysis. Featured device data is aggregated into time windows and then compared to each other and subjective fatigue and sleep related ratings (PROs). General methods to be used are data normalisation, repeated measures of correlation and regressor investigations. These methods can be implemented e.g. using following Python libraries: *pandas, numpy, scipy, pingouin, statsmodels,* and *sklearn*. Finally, the analysis results are typically reported in table or graph format using e.g. the *matplotlib* library.

2 Introduction

The role of the WP4 in IDEA-FAST is to perform device-specific data analysis. The analysis is divided into two parts: A) data analysis aiming to assess the performance of the devices and apps selected for the feasibility study (FS); and B) analysis of the data obtained from devices and apps used during the clinical validation study (CVS). The main goal of the part A was to select the most reliable and accurate devices for use in the CVS. In part B, the goal is to identify and further validate the digital measures of fatigue and sleep obtained from the subset of devices in a larger cohort of participants.

The results of the part A were documented in deliverable D4.3, submitted in the end of September 2021. Devices were organized by four Concepts of Interest (COI) in the feasibility study: Physiology, Activity, Sleep and Social/Cognitive. Analysis was conducted for devices per the COI. The analysis task was divided across four sub-groups of analysts in WP4, each group being responsible for device analysis for a given COI. Bi-weekly meetings together with all the groups ensured that the format and resolution of analysis results were consistent across the groups.

In addition to the data analysis, the aim of WP4 is to provide a software package that allows further development of the analysis methods obtained in the IDEA-FAST project. For the data storage and access, WP5 has developed a Data Management Platform (DMP) where all the data of the project is uploaded and stored. The platform will also include an Analytical Environment (AE) which is intended to run all the analysis steps developed by the sub-groups. This environment facilitates the software package that is being specified in this document.

The primary users of the analysis software package are researchers and data analysts in IDEA-FAST project and future researchers provided with access to the IDEA-FAST data via the DMP. Although results reported in D4.3 provide the foundation of analysis of data from all digital devices, further advanced analysis of the data from the FS is currently in progress. As such, this deliverable is intended to provide the user specification of the AE, leveraging the current analysis framework and pipeline already developed in D4.3. The scope of this deliverable is, therefore, to provide the specifications necessary for users to implement and run the current analysis pipeline, developed in D4.3, on the AE. As new analysis is conducted and methodologies are developed, the technical content of this document is expected to accordingly adapt.

Sections 3, 4 and 5 describe the DMP, data access control and AE in its current state. Section 6 focuses on the data characteristics of each device, suitable pre-processing methods and analytical tools applied for the data. Section 7 describes the methods used in the multivariate analysis. Sections 6 and 7 are based on the analysis conducted with the FS data so far. They aim to provide insight into the memory and processing capacity needs as well as level of complexity in the analysis pipeline.

When moving towards the CVS, it is expected that: 1) data characteristics of the selected devices will remain approximately the same as in the FS; and 2) the quantity of the data will be around 10 to 15 times as much per device compared to the FS, based on the current study plan. This will provide more statistical power to the analytics and most probably improve the prediction accuracy of fatigue and sleep.

3 Platform and environment (ICL)

Device-specific data collected during the project will be stored and managed on the IDEA-FAST DMP. The data analytics software package which contains algorithms for identifying and analyzing device-specific digital measures from the data will be run on the analytical environment of the DMP.

The AE is a secure and user-friendly environment where users can explore and analyse the datasets hosted on the DMP through a web interface. The implementation is based on the Open OnDemand.

Open OnDemand is an open-source high performance computing (HPC) portal funded by National Science Foundation (NSF). Open OnDemand offers an easy way for system administrators at ICL to provide web access to ICL's HPC resource, including:

- Graphical desktop environments and desktop applications
- File management
- Job management

Most of the popular scientific applications are available to users via Open OnDemand, including:

- Jupyter
- MATLAB
- Tensorboard
- RStudio

An overview of the architecture of the Analytical Environment is shown in Figure 1.

- Users use the AE to interact with their remote computing resources through a web browser.
- The AE management layer serves file management and job management services. It also proxies the scientific applications from the computation layer.
- The AE computation layer provides computing resources, which is where the analytical scripts and scientific applications are being run.

Users of the DMP will be able to access the analytical environment to write and execute their own custom scripts to perform analysis based on their needs. The analytical environment will allow users to analyse the project datasets without downloading and keeping a copy of the dataset on their local computers, which further prevents data breaches.

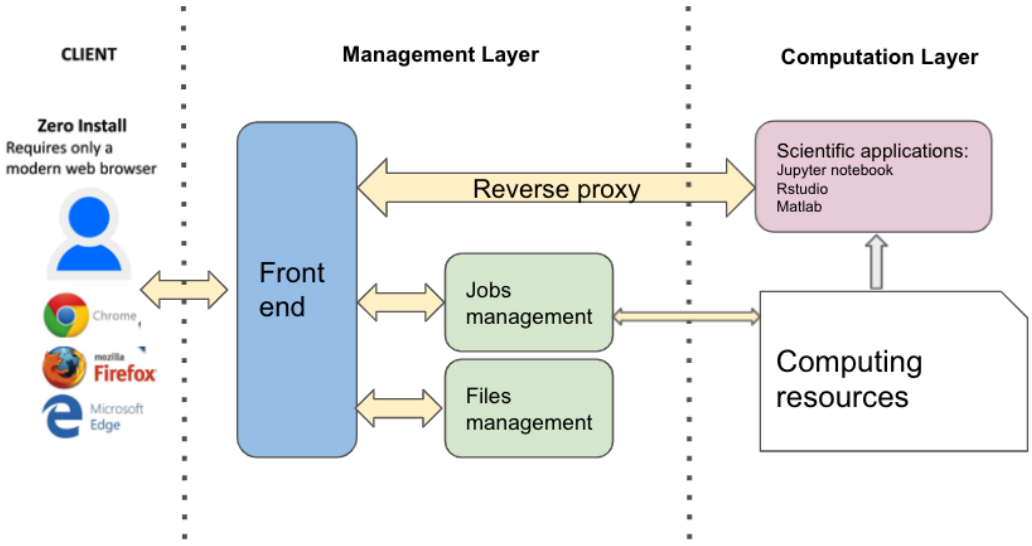


Figure 1. The architecture of the DMP's analytical environment.

4 Data access and governance

The IDEA-FAST DMP and the AE are designed to provide efficient and secure mechanisms to cope with sophisticated data privacy legislations and requirements for data management, in particular, to comply with the GDPR. Authentication and authorisation mechanisms are being designed and implemented to verify users' identities and manage access rights and privileges to data resources. This is described in more detail in D5.2 and summarized below.

4.1 Data Access through DMP Web-based UIs and APIs

The DMP has implemented a Web-based User Interface (UI) so that end-users can access the DMP to transfer data to/from the DMP in a secure manner. Besides the Web-based UI, we also provide APIs so that third-party system and applications from other partners (e.g., clinical eCRF and patient-facing applications) can interact with the DMP to access the resources. Such APIs provide flexibility and convenience for the development and integration of different third-party systems and applications that are being developed in the project. The Web-based UI page provides tools for users to customize their

requests such as upload data and download data in different formats (e.g., raw or standardised formats). There are also several visualization tools providing basic knowledge (e.g., summary of the data obtained). Keyword-based search functions are also provided via the UI to improve data findability and reusability. The implementation of the UI makes use of React1, which is a component-based JavaScript library for interactive UI development.

Applications that interact with the DMP using one of the APIs can filter and search the data by specifying returned fields (e.g., select specific subjects and fields of interest) and adding filters to the data (e.g., return subjects whose weights are lower than a value). The data can be returned either in a raw format or standardized by the IDEA-FAST data standards developed by WP5.

All data querying and manipulation-related tasks are performed via GraphQL2. GraphQL provides flexible access to the underlying data through composition, selection and mutation. Rather than using various path based URIs and HTTP methods, it uses a single endpoint with a predefined schema that specifies how to fetch and change data. Compared to other API design architectures such as REST, GraphQL allows clients to make more precise APIs calls to fetch exactly the data they require from the server, therefore preventing over- and under-fetching.

4.2 Access Control and Data Governance

The DMP allows users to upload data including clinical data, sensor data and general files hosted on the DMP using the Web-based UI or the provided APIs. Users can also perform integrated data modelling and analysis to identify novel digital endpoints via the AE of the DMP.

It is a fundamental premise that no data shall be accessible to any party, internal or external, without full ethical sanction, and that only pseudonymised data will be shared within the IDEA-FAST Consortium. With this access control and data governance in mind, the DMP will implement security and privacy mechanisms that enable users and applications to access the pseudonymised clinical and device data for data exploration and analysis in a secure manner.

Generally, datasets generated in the IDEA-FAST project as well as extant datasets provided by academic and EFPIA partners contain sensitive information (e.g., a participant's clinical records). To ensure data security and privacy, as well as to cope with sophisticated data privacy legislations and requirements, the DMP is designed to provide efficient and secure data access and management. Advanced authentication and authorisation mechanisms with role-based access control model are being designed and implemented to verify users' identity and manage access rights and privileges to data resources. In addition, for data provenance and accountability, an automated audit trail functionality is implemented that tracks all queries/analyses performed against the DMP.

4.2.1 Authentication and Authorisation

Authentication is the process of validating a user's identity to permit access to the DMP. During this process, the user's credentials (such as username/user ID and password) are checked and verified by the DMP. In the current version of the DMP, a Two-Factor Authentication (2FA) mechanism has been implemented to further secure access to project datasets.

Authorisation is the process of determining whether the authenticated user has access to a particular resource on the DMP. During this process, the authenticated user's rights and permissions are checked and verified by the DMP. In the current version of the DMP, Role-based Access Control (RBAC) has been implemented to secure access to project datasets.

1 https://reactjs.org/
2 https://graphql.org/

4.2.2 Access Control and Data Provenance

In the DMP, permission of user to data access control is based on RBAC. A system administrator can assign system-level roles to users, and a system admin/data manager can assign resource-level roles to users to manage access of a specific dataset. Table 1 summarises different user roles and their rights in the DMP.

Table 1. Different user roles in the DMP.

Role	Rights
	System-level User Roles
System Admin	Log in to the DMP, view the list of users, view and modify account information, deactivate/reactivate a user, delete a user account, create and delete a dataset, access datasets on the DMP, access user activity log
System User	Log in to the DMP, access datasets based on permission
	Resource-level User Roles
Data Manager	View the dataset, assign resource-level roles to users, manage access control for the dataset, upload data to the dataset, download data from the dataset, delete data in the dataset, run analysis on the dataset in the DMP analytical environment
Data Uploader	View the dataset, upload data to the dataset, run analysis on the dataset in the DMP analytical environment
Data Downloader	View the dataset, download data from the dataset, run analysis on the dataset in the DMP analytical environment
Data Viewer	View the dataset, run analysis on the dataset in the DMP analytical environment

We have also implemented a logging mechanism so that all user activities on the DMP are tracked and recorded in an activity log. Malicious activities (e.g., suspicious logon and logoff attempts) can be identified which can reduce security risks and prevent data breaches. An activity log record contains information including the username, operation type, time and the request status. Note that the user activity log is only visible to the system admins.

5 DMP Analytical Environment user specs

The DMP AE provides a web-based UI for users to access remote computing resources. Users can access the analytical environment via web browsers with their email address and password.

The AE provides an interactive file explorer. Users can view, edit, upload and download files or scripts in the analytical environment. The AE also provides an interactive jobs submission system. Users can submit their custom scripts to remote computing resources and get computing results back.

Python and R packages are pre-installed in the analytical environment. The packages encapsulate the data APIs of the DMP. Users can access the DMP data via these packages.

Scientific applications are also pre-installed in the AE. These applications can be launched by users and they will be running in remote computing servers. Users can interact with the applications via the web browser. Currently, the AE supports Matlab, RStudio and Jupyter notebook.

For more information, see the user guide of the analytical information in Appendix 1.

6 Device specific data analysis

The data analytics will follow the general analytics pipeline depicted in Figure 2. The pipeline can be executed for individual subjects or used to summarize results (starting from the quality assessment block) over several subjects selected, e.g., by cohort. The correlation analysis can consume data from other sources, such as other devices or the patient reported outcomes (PROs). The analytics pipeline will be executed automatically to a certain extent but, importantly, execution with custom parameters will also be enabled.

Specifically, the analytics pipeline requires access to: (1) all data collected by different technologies; (2) the medical background information of the participants; and (3) patient reported outcomes. The data from all technologies should be linked to the correct study, participant, and device identifier, so that the data can be adequately filtered, as described in section 4, e.g., for cohort analysis. Participant IDs are moreover used to fetch the participant's medical information, such as their disease cohort. Furthermore, when developing the pipeline, the data was consumed in the aforementioned ID-driven hierarchical order (study ID being the top level). Therefore, the pipeline input and output routines expect a similar hierarchical storage structure. The operability of the pipeline depends on the input and output utilities; however, it is possible to adapt them to work in the DMP analytical environment. In addition to the data input, the pipeline also consumes parameter input for the analytics. Importantly, the input parameters need to be adjustable in order to enable different types of analyses. For instance, subject-level analysis could be different from COI-level analysis.

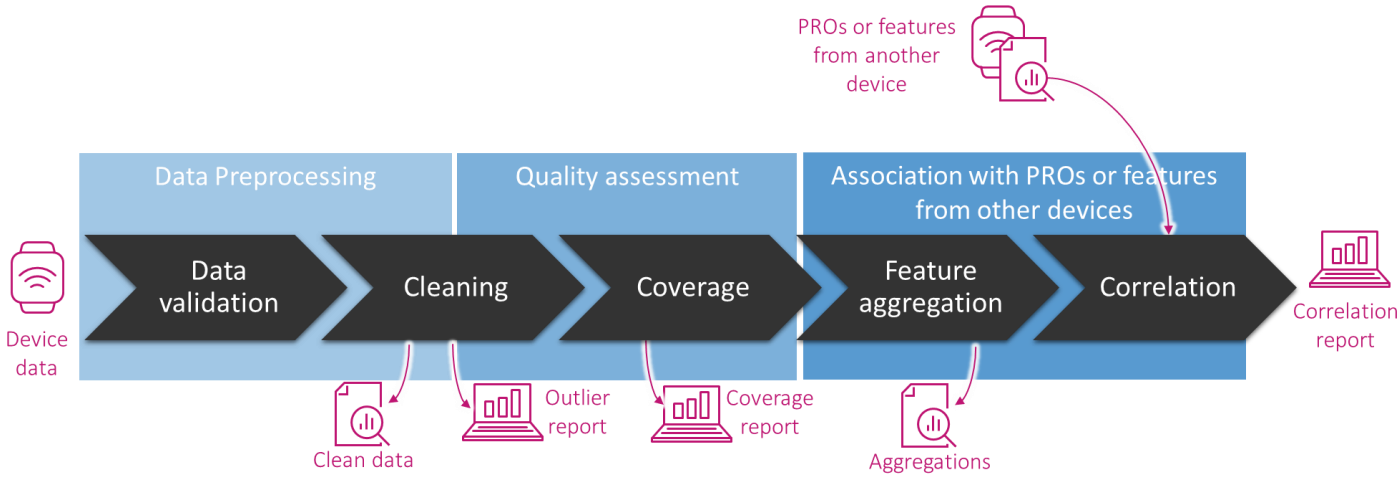


Figure 2. A schematic representation of the analytics pipeline structure, consisting of three analytical blocks.. The most important intermediate outputs (clean data and aggregations) are presented with file icons in the lower part of the figure, and the output reports are presented with laptop icons.

The different steps in the analytics pipeline have some specific requirements. After gaining access to the desired (subset of) input data, device-specific requirements on the data files need to be considered. For instance, the pipeline does not support nested zip files that, e.g., the VTT Stress Monitor Application provides. Inconsistent filenames and typographical errors can impose fatal problems as well, and more strict data upload requirements may come into question when automatic solutions cannot be implemented. Data pre-processing, including validation and cleaning, in general comprise many device- and COI-specific analytics parameters that are further discussed below.

For coverage, feature aggregations, and association analysis, one of the most important general requirements is the adjustable analytics window. The window may be defined via entirely customized window boundaries, or via a predefined length and a single timestamp per window, defining either the start, end, or centre point. Typically, such information is read from a file (e.g. the PRO timestamps).

Different COIs naturally require different aggregation functions that are applied on the windowed data. Bringing the analysis to the final steps, the feature aggregations and the corresponding coverage values need to be accessed together because a coverage threshold is typically used to filter the aggregates before association analysis.

Finally, the analysis results from a selected (sub-)set of participants and devices are summarized. The summary can for instance represent coverage, quality, or association results within the selected group, constructed from the quality results or features obtained for individual participants. The association analysis typically studies similarities between two devices (device-to-device agreement) or between a device and PROs. An example of result visualisation is presented in Figure 3.

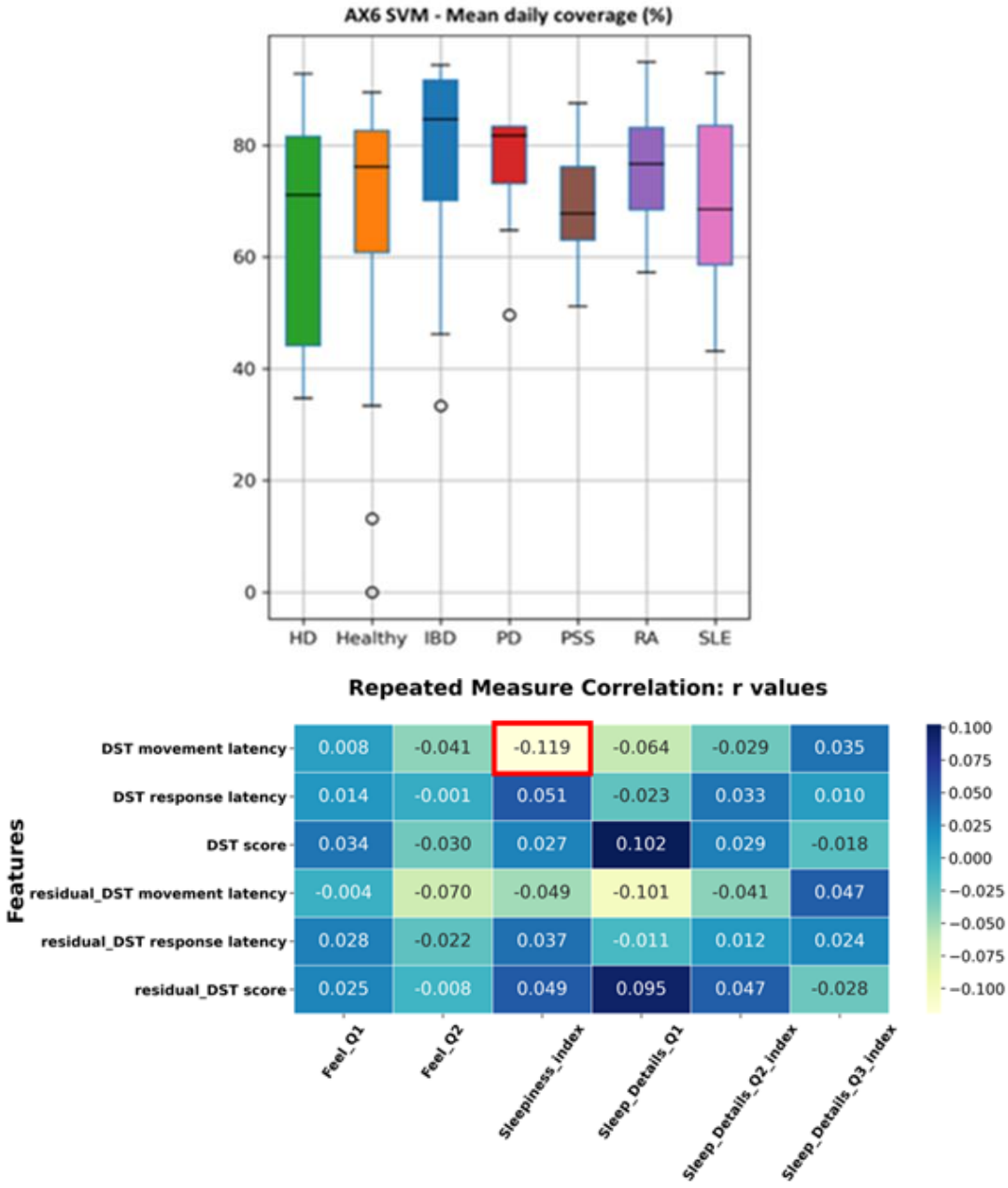


Figure 3: Example of result visualisation: (Top) coverage of the AX6 activity monitor by disease cohort (colours); (Bottom) correlation (R-values in colour-scale, significance indicated with red border) between device data (DST results) and PRO (SMA features).

Naturally, the analytics pipeline may evolve throughout the project, with new code introduced to the pipeline. The pipeline versioning is handled using GitHub. As such, different pipeline versions need to be importable into the DMP AE.

6.1 PROs via Stress Monitor Application

The questionnaire results collected via the VTT Stress Monitor Application (SMA) are embedded in the same data files as the phone usage data collected passively. Because the questionnaires are prompted at predefined times, the coverage and validation process may differ slightly from those technologies with frequently sampled data. However, the low questionnaire recurrence produces rather sparse data, which does not impose strict specifications on the analytics environment, even though the questionnaires need to be parsed from semi-structured JSONL format to tabular format. The questionnaire response extraction only requires the nested zips (SMA data saving format on the DMP) to be extracted, which can be executed programmatically given the generally specified ID-driven hierarchical data structure. The nested storage format may be resolved by separating the audio questionnaire content into separate files in the DMP, which should yield two un-nested zip files.

6.2 Physiological data

For the physiological COI, data is collected with wearables used round the clock. In the FS, the COI-specific devices were VitalPatch and Byteflies (the ECG-dot). Both devices are worn on the thorax.

6.2.1 Data characteristics

The sampling frequencies of physiological data recorded in the FS are listed in Table 2. The raw electrocardiograms (ECG) represent the densest physiological data. In addition to the frequently sampled data, VitalPatch also records R-to-R intervals and Byteflies derives R-peak occurrences in time. Naturally, the infrequent interval data may occasionally be processed differently as compared to other data, for instance when computing data coverage.

Table 2. Physiological data sampling frequencies.

Device	**Sampling frequency (Hz)**			
	ECG	**Heart rate**	**Respiratory rate**	**Skin temperature**
VitalPatch	125	0.25	0.25	0.25
Byteflies	250	0.1	1/60	-

6.2.2 Reading and pre-processing device-specific data

The two devices for physiological measurements require device-specific pre-processing, which may however be possible to execute automatically as the data arrives. For VitalPatch, the ECG-derived features and skin temperature are scattered across multiple files and need to be parsed together for each participant. The data columns need to be named retrospectively and Unix timestamps interpreted as datetime. For Byteflies, the data is not only scattered into multiple files per participant, but also in different files by data type. The data is first processed by WP3. Thereafter, fetching the correct data requires scanning through the metadata to locate the desired files. Importantly, the measurement start time needs to be read from the metadata and added to the duration in the data files to yield the correct datetime. Once the data are collected together, it is convenient to name the columns consistently to what is expected by the analytics pipeline modules by default. Finally, the Byteflies R-peak data should be processed into R-to-R intervals.

As for the raw ECG data, advanced analysis from either device may require, e.g., detection of specific parts of the ECG signal, for instance the QRS complex. For QRS detection, computational complexity is at least linear but can be notably higher for more elaborate algorithms (e.g., wavelet-based detection).

Physiological data cleaning accounts for non-physiological values (range outliers) and contextual outliers. The latter are localized using a sliding window of predefined length. In addition, each device may require specific cleaning steps. For instance, VitalPatch data needs to be inspected for invalid values (one predefined value for each raw signal or derived feature), whereas Byteflies incorporates metadata with a quality label for the full recording as well as labels for individual R-peaks that need be considered upon cleaning. Furthermore, if the raw data quality was insufficient, the derived features may exceed the expected sampling frequency. Hence, such sequences should also be cleaned from the data.

6.2.3 Analytical interests

The length of analytical windows for physiological data are usually driven by heart rate variability (HRV) analysis. One of the most typical ways is to perform HRV analysis over one night's sleep (several hours), but some applications may use much smaller windows (such as 10 minutes), although it affects the interpretation of the results. And because physiological parameters strongly interplay with physical activity, the activity analysis can also complement the physiological analysis, indicating some specific windows of interest (e.g., recovery after exercise). On the other hand, it may be interesting to inspect also much larger windows, e.g., daily or weekly resting heart rates. Naturally, aggregations over longer windows produce less data, reducing the need for computational resources.

6.2.4 Resource requirements

The most resource consuming part in processing physiological data presumably arises when processing the raw ECG data. For instance, for VitalPatch, individual files exist in the order of tens of megabytes, and each participant produced hundreds of such files already in the FS. Combining the high volume of data with potentially highly complex algorithms can explode the processing time. This may be helped by performing the processing in parallel for individual files, before combining the results to create one resulting file for each participant. For ECG-derived features, data cleaning is one of the most demanding steps, namely filtering outliers with sliding windows. For small analysis windows, extracting the HRV parameters using the hrv-analysis package (https://github.com/Aura-healthcare/hrv-analysis) may also become demanding.

For advanced analysis, implementing machine learning and neural networks for data cleaning or analysis will likely require multiple CPUs running in parallel and, most importantly, GPU or TPU resources with large memory (at least 11 GB for e.g. Transformer networks).

6.3 Activity data

Within the Activity COI, we had four devices in the FS attached to various body locations. For example, MoveMonitor (MM) was attached on the lower back, Axivity (AX6) was attached to the wrist, while the Byteflies (BTF) and VitalPatch (VTP) were attached to the ankle and chest respectively. Each device has either built-in accelerometer (Acc) or gyroscope (Gyr) modules e.g. MM, AX6, and BTF have both Acc and Gyr while the VTP has only an accelerometer.

6.3.1 Raw data

The more modules each device has, the more space/storage capacity we will need. In addition, the raw data file format also plays an important role e.g. MM and AX6 have raw file format in .OMX and .CWA respectively which takes less than 1 GB of storage for one time assessment (around 7-10 days), while the BTF and VTP store the data in the .CSV formats which can take more than 5 GB of storage for each individual subject. Furthermore, each device can send the data at different frequency rates configured by the assessor during the clinical assessment. The approximate sampling frequency of each device is given in Table 3.

Table 3. Physiological data sampling frequencies.

Device	**Sampling frequency (Hz)**	
	Accelerometer (Acc.)	**Gyroscope (Gyr.)**
MoveMonitor (MM)	100	100
Axivity (AX6)	100	100
ByteFlies (BTF)	25	25
VitalPatch (VTP)	Dynamic	---

6.3.2 Reading of raw data and filtering

Reading the raw data can be computationally heavy as the code will read the 7-10 days of data recorded at a certain frequency for saving it into .CSV or .MAT file or perform certain operations on it by loading a chunk of it. For efficient processing of the MM and AX6 data, the code provided by the "Open Movement" (https://github.com/digitalinteraction/openmovement-python) can be beneficial.

On the other hand, raw data from these devices does not always come at the same frequency. So data interpolation (up sampling or down sampling) can be computationally heavy to match the data rate for every second. After proper sampling of the data, data filtering to remove the unwanted noise is generally a less computationally heavy step, python library "SciPy" can help here or built-in "Signal Processing" toolbox in Matlab can also be utilized.

6.3.3 Implementation of the algorithms for feature extraction

Custom algorithms developed in Matlab and Python will be utilized. The folder structure and file naming convention will remain the same even if a different software/platform will be utilized. Features from the raw accelerometery data will be extracted day by day so running multiple blocks of code for activity classification first such as gait/turning identification and later extracting detailed clinically relevant features for analysis. For this purpose the published algorithm will be utilized e.g. GaitPy (https://joss.theoj.org/papers/10.21105/joss.01778), OpenMovement (https://github.com/digitalinteraction/openmovement-python), and Accelerometer from UK BioBank (https://biobankaccanalysis.readthedocs.io/en/latest/methods.html/, https://github.com/activityMonitoring/biobankAccelerometerAnalysis). Furthermore, these activity devices can also be used for the sleep/time on bed estimations as the subjects wear these devices for a complete 24 hours. Therefore, the acceleration signal can be used for this purpose. An open source package in R named as GGIR (https://cran.r-project.org/web/packages/GGIR/vignettes/GGIR.html) is frequently used in the literature for sedentary activity behaviour (severe, moderate, vigorous) and for objective sleep quantification from a wearable inertial sensor data. Reading files by this package and storing them first time can be computationally heavy as it will need to run through all the folders to search for the proper file extension and then extract it.

6.3.4 Summarizing data

Until now in the FS, we have not used the custom algorithms for the feature engineering. Instead, we relied on the features provided by the device manufacturer directly. We presume that the data summary will remain the same in both cases: 1) we use the features provided by the device manufacturer; or 2) we use the features extracted with the custom algorithms on the analysis platform. The computationally heavy steps are briefly highlighted in the below process, which we have used for the FS data analysis.

a. Data Validation/cleaning

The features summarized as epoch by epoch (1 minute length) are used in this process. If the features provided by the device manufacturer do not have regular intervals then re-arranging the data can be computationally heavy. As we had this case only for the MM device, the rest of the devices provided the features at proper intervals. Processing of the BTF data to have the activity features from the ankles also increases the workload due to finding the proper labels from the JSON files. For the VTP device, further data cleaning to remove outliers or to merge data from different days are not computationally heavy.

b. Data daily coverage and quality

The function used for the daily coverage and quality assessment are efficient and not computationally heavy. One thing to keep in mind here is to adjust the percentage of good quality data for keeping a specific day for analysis. In addition, the number of days used for the coverage in the current analysis was 10. Perhaps in future we may need keep this number dynamic rather than fixed. However, these points will not affect the computational time.

c. Matching with PROs – Window size

Within the FS analysis, we have observed that window size can have impact on the association between the PROs and the features provided by the device manufacturer. Therefore, keeping this number dynamic in the code is important. As the code will be running on the already processed data, this step will not be computationally heavy.

For this step, PRO data should be in the separate folder already extracted in the .CSV file.

d. Saving data

At every step mentioned above, the output will be written to a .CSV file within the subject/device folder or within the subject folder but outside of the device folder, depending on the type of output. The final data summary will be saved within the study folder along with the plots of coverage and data quality.

6.3.5 Other computationally heavy methods

Activity classification methods using machine learning (deep learning) methods can be computationally heavy. For example, training a deep neural network such as CNN or LSTM on the raw time series data will require computational power supported by the GPU/TPU. The other computationally intensive tasks can be to extract signal-based features directly from the timeseries data to find new markers for better predication/correlation with PROs. Therefore, appropriate mechanism within the analytics tool is required to allocate appropriate resources automatically or assign some labels to request resources while submitting the job to the platform.

6.4 Sleep Data

For the sleep COI, three main devices have been considered for measurements in the FS. Specifically, eBedSensor is a force-sensitive piezo-electric film that is placed under the mattress during sleep. ZKONE is a wireless sensing radar device which uses an Ultra-Wide Band (UWB) signal to detect human vital signs. The Dreem headband is worn by the subject during sleep and records physiological data in real time. All three devices are able to provide basic sleep indicators and the corresponding measured data can be collected through either a physical portal or dedicated cloud platforms.

6.4.1 Raw Data and Feature Extraction

For the three sleep sensors, some data processing algorithms are ether built into their hardware or deployed in a remote server. After evaluating the nocturnal recordings from participants, basic physiological measures (e.g., average heart rate, average respiration rate) and sleep-related assessment results (e.g., sleep duration, hypnogram, and sleep efficiency) are stored in '.JSON' files (for ZKONE and Dreem) and '.CSV' files (for eBedSensor). It is worth noting that the raw data format of eBedSensor is represented by '.HRV' files, which can be fed to an internal pipeline to generate the above aggregated '.CSV' files for further analysis.

In order to extract relevant features from raw data, while taking data format unification into account, the .JSON files from ZKONE and Dreem devices are first processed by extracting and aligning features from all given participants. The intermediate .CSV files could be saved for each device category. In detail, some identifiable information are marked in the file name. For instance, a .CSV file from the Dreem device could appear as "E3C7X5F-DRM7HH5EN-20210407-20210505_aggregation_12.csv", where "E3C7X5F" is the participant ID, "DRM7HH5EN" is the device ID, followed by the starting and ending dates for the data recording. The last number "12" means the effective days for data collection. Thus, all saved files would be treated as input to conduct the following coverage statistics as well as correlation analysis.

6.4.2 Data Cleaning

After feature extraction, data cleaning is deployed to filter out abnormal nocturnal recordings for every participant. Note that duplicated recordings belonging to the same night should not be counted. So, in our pipeline, duplicated recordings are discarded by keeping only one recording with longest sleep duration per night.

After that, to define outliers for each device, a 2-hour sleep duration is utilized as a normal threshold to pick out too-short sleep episodes. Specifically, for ZKONE and Dreem, the parameter 'sleepduration' is utilized, while the parameter 'SLEEP RECs' is utilized for eBedSensor. The information of all outliers will be stored in the sub-sheet ('Outlier_sleeps') with participant IDs and testing dates listed in detail.

From the patient database provided by UCAM, intended device testing periods are utilized to calculate the nightly coverage for the participant and cohort information is utilized to split coverage results into different sub-sheets of the same summary Excel file for each device.

6.4.3 Data Mapping for Correlation Analysis

For the purpose of studying associations with PROs or features from other devices, we adopted an aggregative mapping strategy which selects the most desirable features from the three sleep sensors. To map with PROs for each participant, the time basis falls on the PRO's testing duration. To map within the sleep COI category, we stick to the shared testing duration for all devices. After the mapping for all participants, an additional grouping step is implemented based on the UCAM cohort

information. Finally, as for the feature unit unification issue, we also standardize all the time units into seconds and convert all percentage parameters into the interval of 0 to 1.

a. Mapping with PRO

In practice, there exist lots of manual input errors for the PRO application, e.g. 'To_bed_Time', which might result in usually a 12-hour time shift. Additional time calibration or compensation is needed in such cases before the mapping procedure with PROs. This sub-function has been built into the pipeline. Likewise, the aggregated device features along with PRO counterpart would be saved as .CSV files for the future correlation analysis.

b. Mapping across Devices

As mentioned above, we combine all three sleep sensor features according to their shared recording duration for the study of cross-device agreement. Similarly, they are also saved as intermediate .CSV format. The selected features are identical to those for the PRO mapping.

6.5 Social data

Social data for the Feasibility Study were collected, together with the patient reported outcomes (PROs), using the VTT Stress Monitor App. To date, there are 147 data files available in the IDEA-FAST DMP containing SMA data, which comprise around 1.2 GB of data volume. As we mentioned before, these files are also processed within the other COI pipelines to extract questionnaire answers. Raw social data are stored in JSONL files, which contain mobile data records including a timestamp, a data type (e. g. PHONE_SCREEN in the case of screen on/off related records), a data value and a geolocation code. As these instances are recorded when changes occur in the mobile phone (i.e. the patient turns on/off the screen, the phone battery changes, the patient completes a questionnaire…), social data do not have a fixed sampling frequency. Thus, the analysis conducted on social data mainly consist of selecting records based on the instance type to compute features within fixed timeslots.

6.5.1 Raw data pre-processing

Following the general analytic pipeline of Figure 1, raw social data should be pre-processed before conducting any coverage/feature extraction process. The pre-processing steps include missing data removal (i.e. files with no data recorded or without questionnaire answers[3]), validating the timestamp of the records, removing files with few data recorded, or merging the data of patients with multiple files. These steps can be conducted using common data science Python libraries like *Pandas* and *NumPy*.

6.5.2 Data aggregation and feature extraction

During the FS, social data were aggregated in timeslots of 24 hours (daily windows). The analysis conducted by WP4 revealed that there is no significant difference in the results when using time windows of both longer (2-day) and shorter (8-hour) duration. Unlike other COI data, social data usually presents large periods between samples of interest (i.e. those related to screen activations and foreground apps). This fact led us to discard short-time aggregation windows to maximize the utility of the information in each window. For example, choosing a short time window would reduce the number of screen activations within each timeslot, thus limiting the variability in the features extracted.

3 A list of IDs with empty SMA files or with no questionnaire answers can be found here. However, there are code implemented in the IDEA-FAST GitHub to detect and remove these files as a preprocessing step of the SMA data.

On the other hand, using long-time windows reduces the number of data points to compute the correlation with the PROs.

Attending to the feature extraction steps, we propose to compute several screentime statistics (e.g. total screentime, number of screen events or median time per screen activation), and the frequency of each app category within timeslots. Again, due to the dataframe-like nature of social data, feature extraction can be easily done using both *Pandas* and *NumPy* libraries. When extracting screentime features, it is important to remove unmatched PHONE_SCREEN records, that is, screen instances without its corresponding pair. For example, if the first screen instance in a record is a SCREEN_OFF notification, or the last one is a SCREEN_ON, we cannot extract any screentime from those notifications. The same applies when we encounter two consecutive PHONE_SCREEN instances with the same value. Another case to consider are extremely long screen activations, which are normally due to application breaks and gaps in the data.

During feature extraction, we also recommend computing the mean and the standard deviation of each feature to remove outliers before associating the features with the PROs.

6.5.3 Correlation Analysis

Before starting the correlation analysis, it is important to extract questionnaire answers and store them in a separate .CSV file. We note here that 'Activities_Q3 (Other comments)' is designed as a free, optional answer question, which usually has no response across subjects. Hence, we recommend to not consider it during the analysis. Conducting the repeated measures correlation analysis is probably the computationally heavier step in the social pipeline. While this analysis presents no problems when computing the association between screentime features and the PROs, due to the distribution of the app categories across subjects, the correlation analysis between foreground apps features and the PROs was restricted to only 2 categories during the FS. The correlation analysis can be conducted using Python libraries like *Pingouin*. As there is only one device recording social data, we did not perform any mapping across devices within the social COI.

Due to the lightness of social data, it takes around 10 to 15 minutes to obtain both the correlation and the data coverage results from running the social pipeline on the raw records from all the patients.

6.6 Cognitive test analysis

Cognitive data for the Feasibility Study were collected using the ThinkFast App (TFA). The IDEA-FAST DMP contained 166 files when the analysis for D4.3 was performed, amounting to ~8MB of compressed data volume. The compressed containers were named following the format PatientCode-DeviceCode-YYYYMMDD-YYYYMMDD. Often, more than one container is generated for each subject. The raw data are stored in JSON files, which contain data records including timestamps for start and end time of each session, and a list of items describing each datapoint through fields like "measureCode", "measureDescription" and "Result".

Reminders to take the test were presented to the subjects twice per day, in the morning and in the afternoon, with the goal to acquire measures twice per day. Three types of tests were presented: PVT (for 1 week), DSST (for 1 week), and NBX (for 2 weeks).

6.6.1 Raw data pre-processing

From the numerous measures obtained with the TFA tests, the 8 most relevant were identified with regards to the fatigue COI, and considered as base features. Due to the discreet nature of the TFA data and the relative small size of the files, it was convenient to merge all the entries in one unique file before proceeding with coverage and correlation analysis.

The libraries *zipfile*, *tkinter*, *pandas*, *pathlib*, *os*, *json* and *datetime* were used for the scope.

6.6.2 Coverage analysis

Coverage calculations were computed at TFA level (measuring compliance of the participants) on three different levels: per test type, per disease group, and in total. 100% coverage corresponded to having provided the expected number of answers (i.e. 2 per day, for 5 days per week, per test). It has to be noted that even this was an arbitrary choice, as the reality of the data morphology was more complex (people took the tests for longer periods than expected, or for more than 5 days per week).

The Python packages required to run the code are *NumPy*, *scipy* and *matplotlib*.

6.6.3 Data aggregation and feature extraction

In order to account for possible learning curves in the performance of the tests, for each of the 8 selected features, the corresponding "learning-curve-residual" feature was calculated. This means that if the linear interpolation of the datapoint in time (for each participant and for each feature) had a positive angle, the residuals were computed from this line. On the other hand, if the data in time showed no improvement (i.e. negative or null slope), just the average was removed to calculate the residuals.

With the goal of correlating PROs and TFA features, the data were then aggregated in 24-hours slots. Because of the scarce alignment between TFA and SMA data-points, the choice of shorter time-windows was ruled out because it reduced the chance of both SMA and TFA data being included in the same slot (i.e. causing data loss). Moreover, shorter slots did not qualitatively change the final correlation results. For each participant, one file for the SMA and one for the TFA included the list of daily average of their scores.

The Python packages required to perform these steps were *pandas*, *pickle*, *NumPy*, *scipy* and *matplotlib*.

6.6.4 Correlation Analysis

Repeated measures correlation analysis was conducted on the aggregated PRO and TFA data, derived as described in section 6.6.3, above. Repeated measures analysis was conducted using the *Pingouin* package in python. With the relatively compact datasets involved, the process is not computationally intensive and correlations could be obtained across all cognitive outcome measures and PRO measures in approximately 10 minutes. Consistent with the constraints around analysis discussed in section 6.5.3, only data from SMA PRO questions producing numeric responses were considered.

Python packages used in data import, shaping, analysis and visualisation are: *Pandas*, *matplotlib*, *pyplot*, *NumPy*, *os*, *re*, *glob*, *seaborn*, *openpyxl* and *Pingouin*.

7 Multivariate analysis

The multivariate analysis combines multiple sources of data to build regressors that predict self-reported outcomes. To run the analysis, different sources of data are needed:

- Demographics information is used as covariates and can be found in clinical data (UCAM data). PROs assessed at baseline may also be used as covariates;
- Digital features representing the five COIs that were extracted automatically from raw sensors by WP4 (raw data pre-processing and feature extraction is detailed in section 6);
- Weekly PROs (UCAM data);
- E-diary questionnaires automatically extracted and cleaned from SMA raw data (section 6).

Digital features are aggregated to match the PROs frequency (weekly aggregation to match weekly PROs, daily or more detailed aggregation to match daily self-reported outcomes) and features with low coverage are removed from the analysis. A mapping between participants and timestamps is performed to gather all sources of data: covariates, aggregated digital measures from all COIs, PROs. Datasets may be created on different time windows and match different types of PROs. They are saved in a .CSV file to be used in multivariate analysis.

7.1 General approach

A cross validation procedure is applied that splits the dataset into a training set and a test set. The model is trained on the training set and its performance is assessed with the test set. A "leave-one-out" cross-validation puts one participant aside in the test set. If the dataset is stratified into k folds, the stratification must preserve the percentage of participants from each cohort within each group as much as possible.

A feature selection is then applied on the training set. To remove redundancy between features, digital features are clustered based on their pairwise correlations. Within each cluster, only the measure with the highest correlation with the PRO is kept to build the model.

Features are normalized (z-norm) and missing data are imputed with the participant average value. PRO values are transformed to get closer to a normal distribution (e.g. boxcox transformation).

Different types of regressors are investigated, such as linear, ordinal, non-linear, and mixed-effects regressors. Analysis is performed on the total population or on sub-groups of participants, in particular per cohort or per group of cohorts.

The python libraries required to perform the features selection step are *scipy*, *statsmodels*, *pingouin*. The cross-validation procedure, the performance assessment of built models, and machine learning models training steps are implemented using *sklearn* library. Linear, ordinal, and non-linear mixed models can also be investigated using R packages *lme4*, and *ordinal*.

7.2 Resources requirements

In the cross-validation procedure, the feature selection may be computationally heavy, especially the computation of the pairwise correlation matrix when a lot of features are involved, mostly if repeated-measures correlation is computed. In future analysis, other methods to impute missing data (e.g., model-based imputation) will be used that may require more resources.

For advanced analysis, implementing recurrent neural network directly on raw sensor data will require GPU or TPU resources with large memory.

8 Requirements summary

As sections 6 and 7 indicate, the need for data processing of each COI depends on the characteristics of the data. However, there are some commonalities in the data handling. The data analytics pipeline generally requires a variety of python packages to provide basic functionalities, such as presented in table 4.

Table 5 summarizes the COI-specific data characteristics, features and needed libraries for the data processing.

Table 4. The basic libraries and toolboxes needed in each phase of the analysis pipeline including multivariate analysis and machine learning methods

	Data input	**Analytics**	**Visualisation**	**Other**
Python libraries	os, glob, re, zipfile, pathlib, sys, packaging, json	pandas, numpy, scipy, pingouin, statsmodels, scikit-learn (sklearn)	matplotlib, seaborn	sys, packaging, datetime, dateutil
Matlab toolboxes		Signal Processing, Statistics and Machine Learning Apps		
R Studio	readxl	lme4, ordinal, tidyvers, nlme	ggplot2	Reticulate, foreign, dplyr, DHARMa, olsrr, HLMdiag

Table 5: Summary of COI data, typical features and needed libraries

COI	**Data acquisition**	**Raw data rate**	**Key Features**	**Feature time win**	**Feature description**	**Special libs**
Daily PROs	Prompted	4/day	Fatigue, Sleepiness, Anxiety		Subjective rating	*pytz*
Physiology	Evenly sampled and irregular	125-200 Hz	HR, RespR, R-to-R	1min-Daily	Peak detection, Frequency analysis	*hrvanalysis, (neurokit2)*
Activity	Evenly sampled	25-100 Hz	Steps, Energy, SVM	1min-Daily	Movement magnitude, step detection	*OpenMovement, UK BioBank Accelerometer, GaitPy, tfresh, GGIR (R)*
Sleep	Evenly sampled during night	Beds 110Hz Dreem 250Hz ZKOne 0.5Hz	Sleep Q Sleep time RespR WakeUps	Nightly	Frequency analysis, repeated measure correlations	*itertools,* *collections,* *sqlite*
Social	Phone usage/events	~50 events/day	Screentime appUsage	Daily	Mean screen time, app ID count	
Cognitive	Prompted	2/day	DST Move-ment latency, PVT time-outs, residual PVT time-outs, residual PVT Score	Daily	Residuals from linear interpolation, repeated measures correlation analysis	

9 Appendix 1 – IDEA-FAST Analytical Environment: User Guide

This document provides guidelines for a user to:

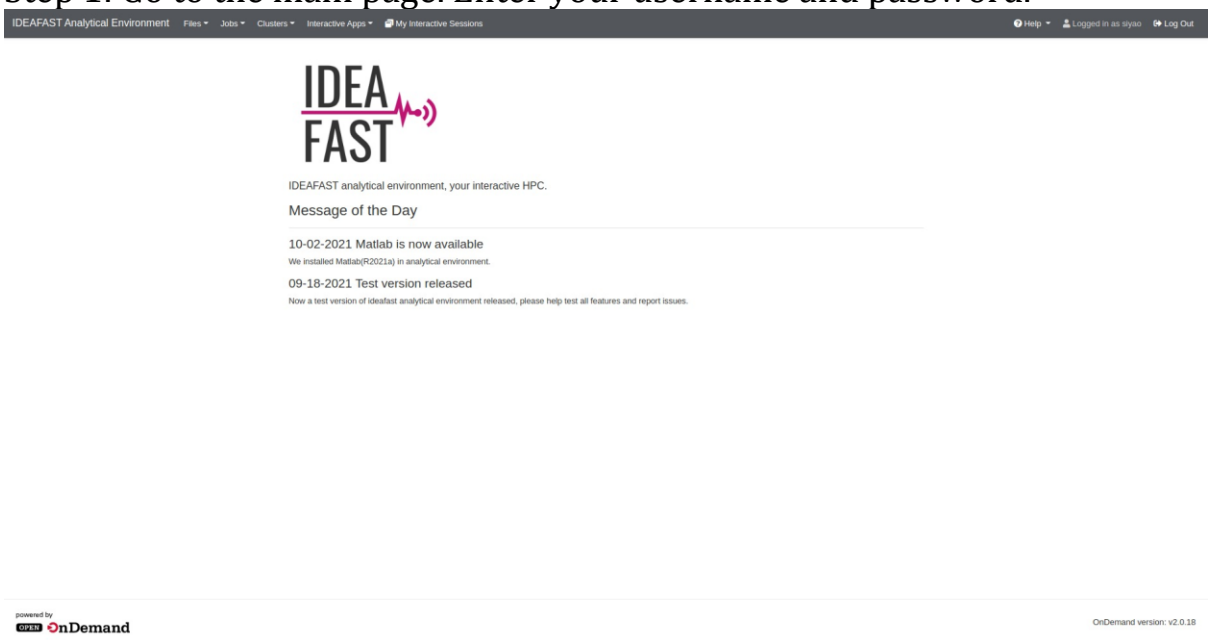
1. Use the IDEA-FAST analytical environment to run customized computing scripts;
2. Use the built-in services of several popular computing tools, including MATLAB, Jupyter and RStudio. A web-based desktop GUI is also provided (Spyder is in progress).

Notice:

1. The Analytical Environment (AE) uses ***dmpy*** to interact with the IDEA-FAST Data Management Platform (DMP). *Dmpy* is a python wrapper of the Application Programming Interfaces (APIs) of the DMP.
2. The Analytical Environment does not provide licenses for those services required (e.g., MATLAB). You need to activate such services with your own licenses (see below).
3. As of the date of this deliverable, the AE has not yet been formally released.

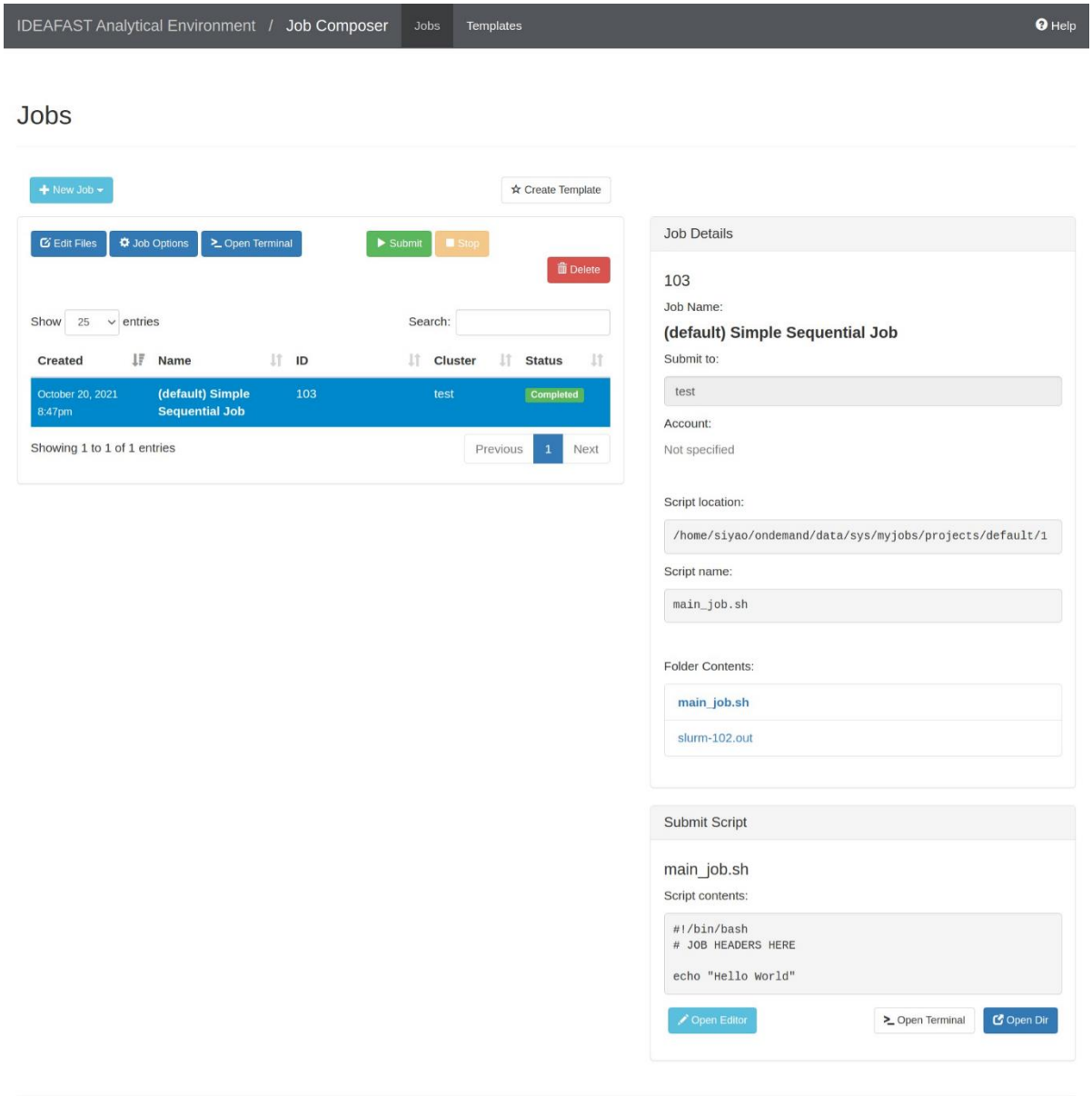
Login

Step 1: Go to the main page. Enter your username and password.



Create a job:

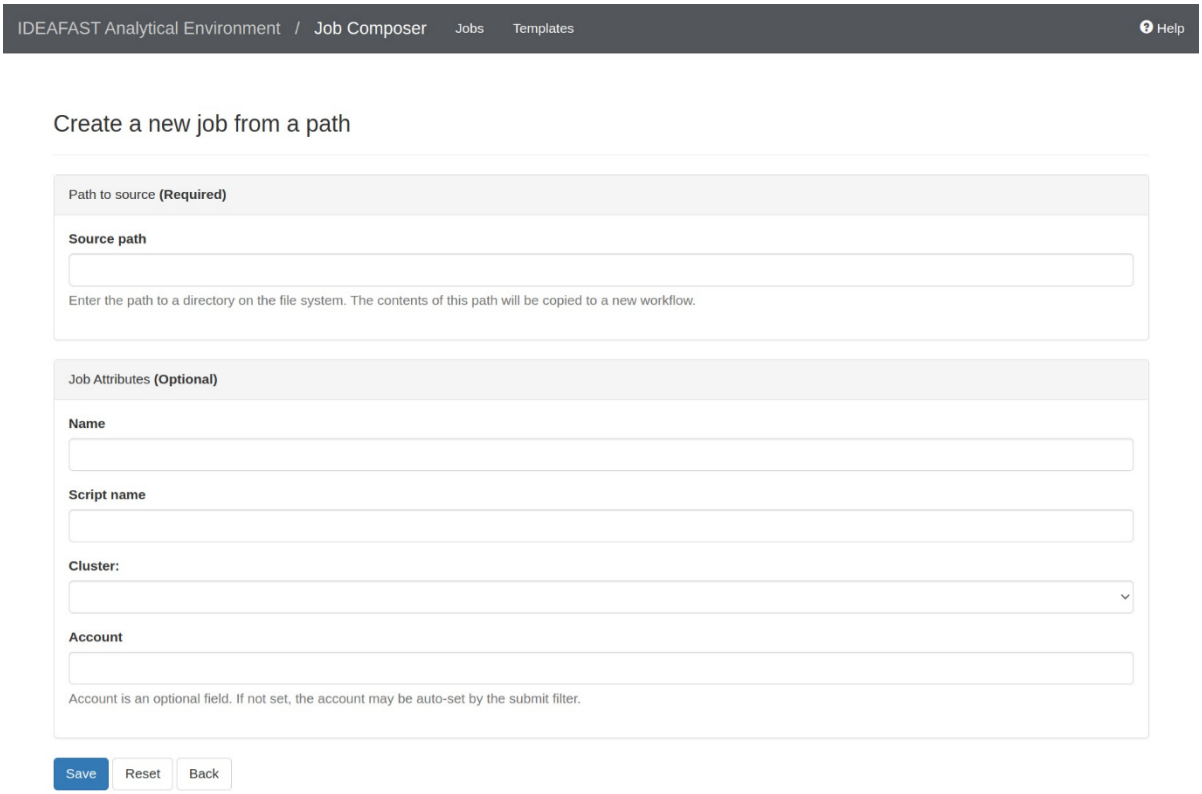
Step 1: Click "**Jobs -> Job Composer**" on the main page:

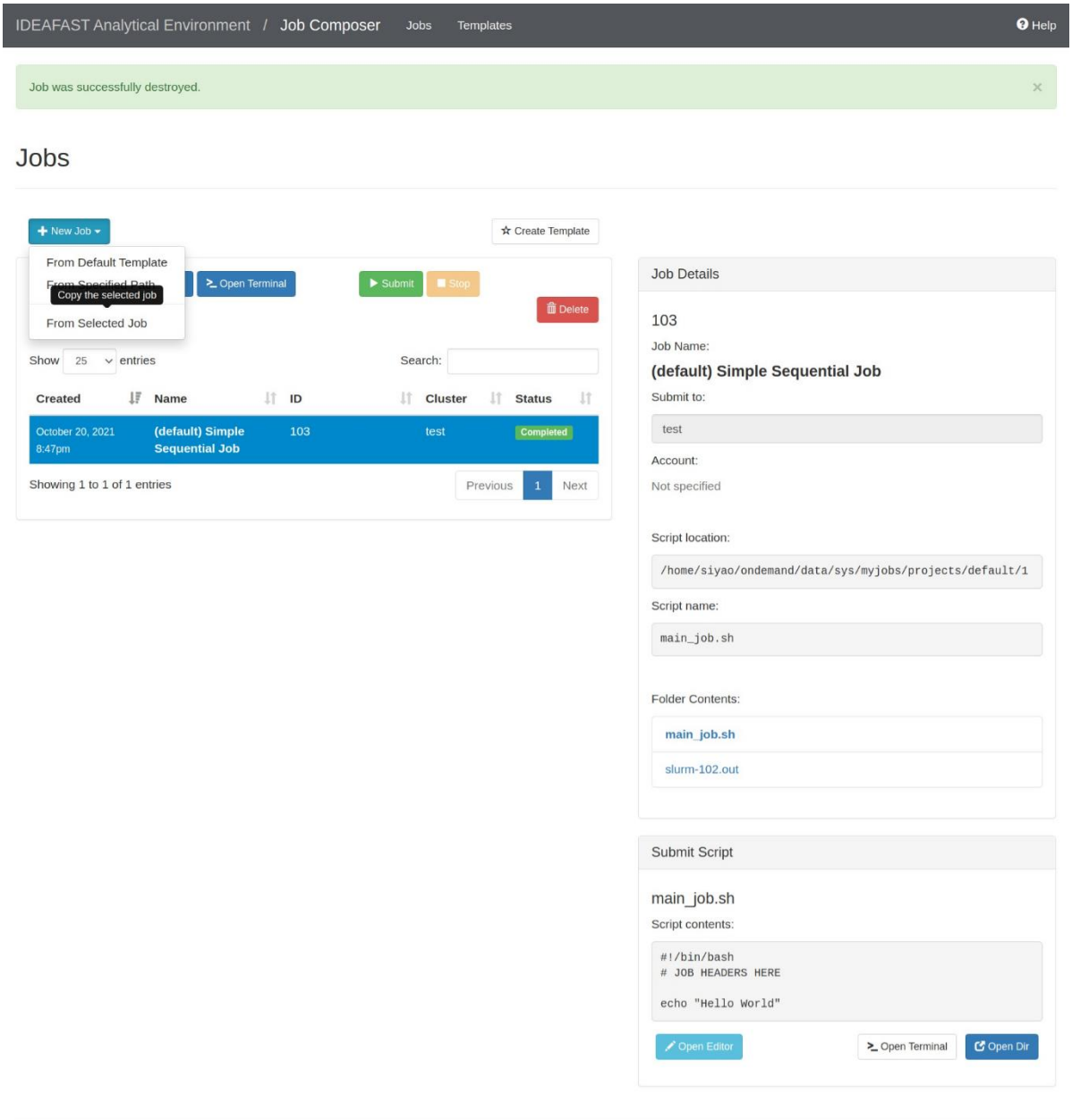


In this page, you can view all submitted jobs and their details.

Step 2: Follow the instructions on this page to create a new job:

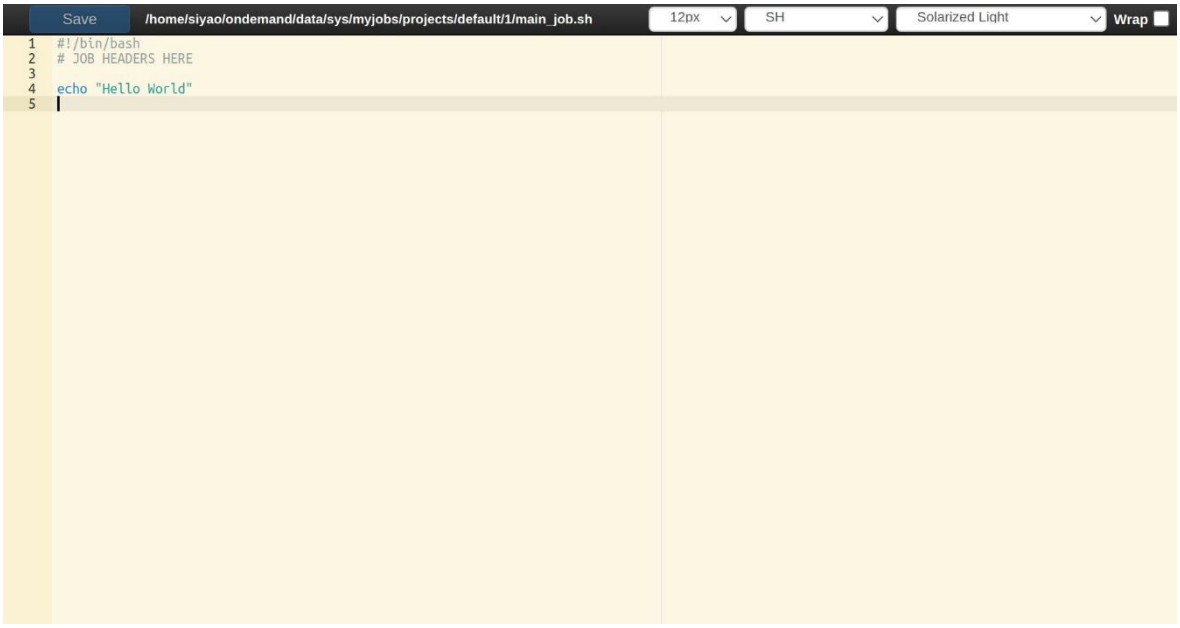
1. Click "**New Job**" to create a new job using one of the three approaches:
    a. From default template: run a default template to test if the AE works well;
    b. From specified path: you need to specify the parameters of this job, see the figure below;
    c. From selected job: rerun an old job. To use this approach, you need to select an old job from the job list first then create a new job. See the figure blow.


IDEAFAST Analytical Environment / Job Composer
Jobs
Templates
Help
Create a new job from a path
Path to source (Required)
Source path
Enter the path to a directory on the file system. The contents of this path will be copied to a new workflow.
Job Attributes (Optional)
Name
Script name
Cluster:
Account
Account is an optional field. If not set, the account may be auto-set by the submit filter.
Save
Reset
Back



You can also customize your own template for further use.

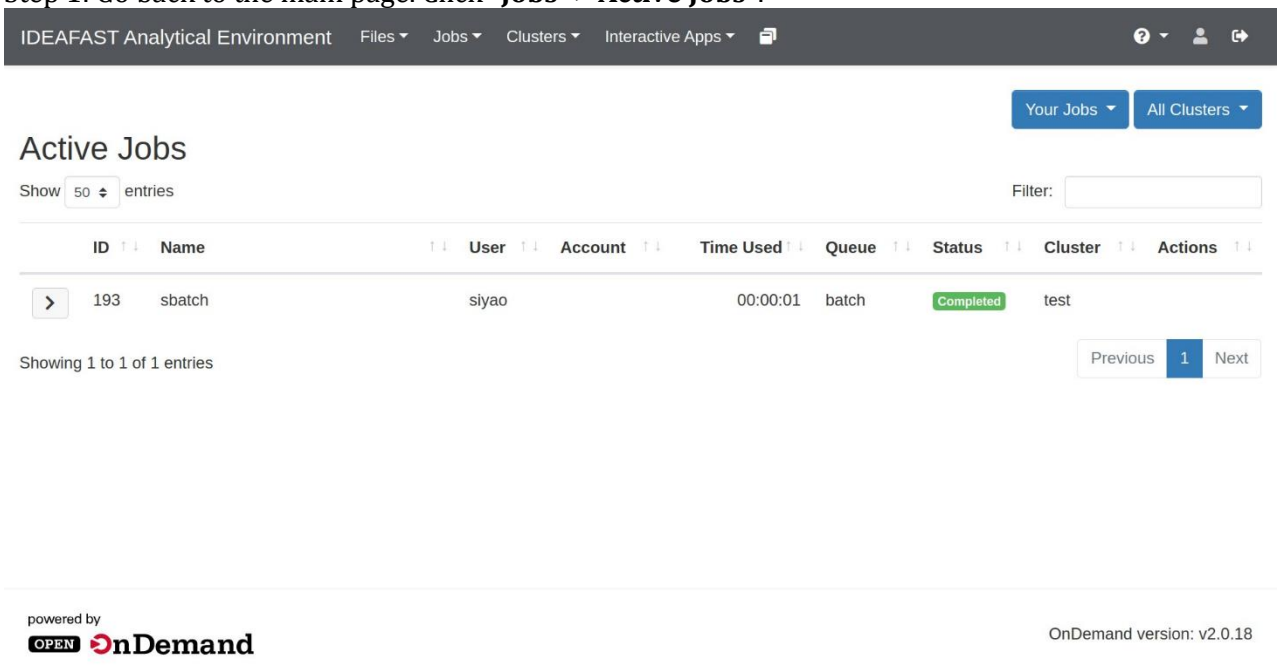
2. Generally, you are not supposed to edit the scripts at this page; but we provide the online-editing tools for you if necessary. Click **'Open Editor'** at the bottom of job details to make any changes.



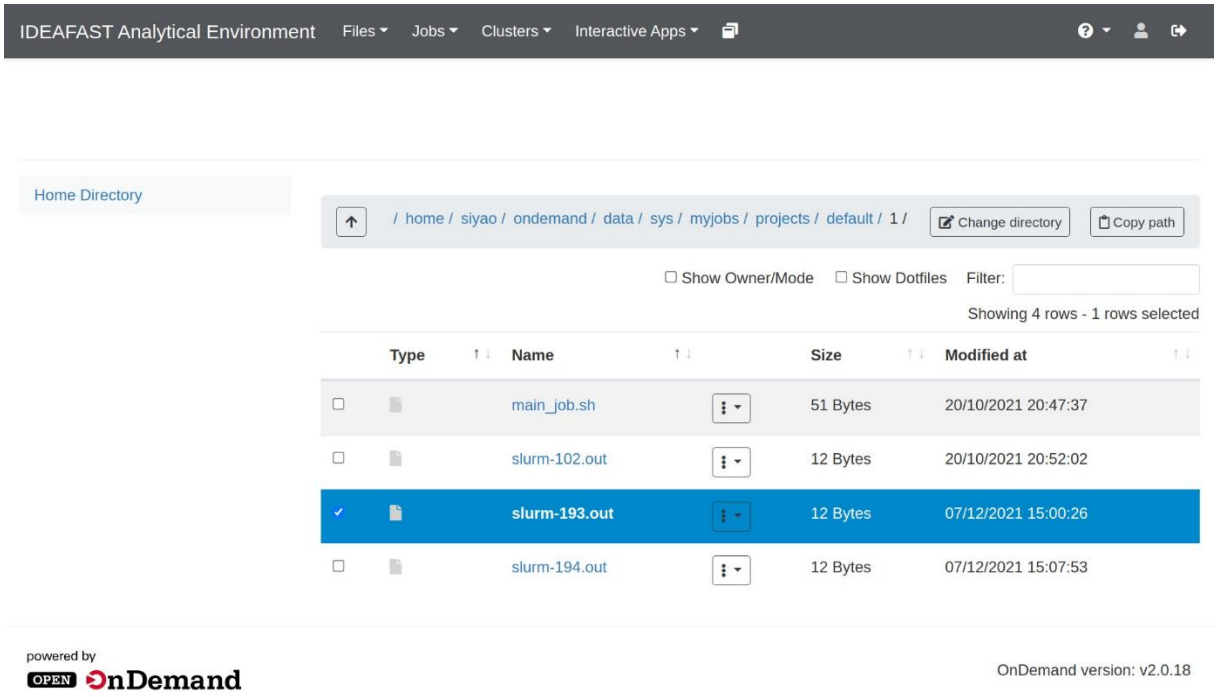
3. Go back to the job page. Select the job and click **"Submit"** to submit a job.

Check the status/results of a job:

Step 1: Go back to the main page. Click "**Jobs -> Active Jobs**":

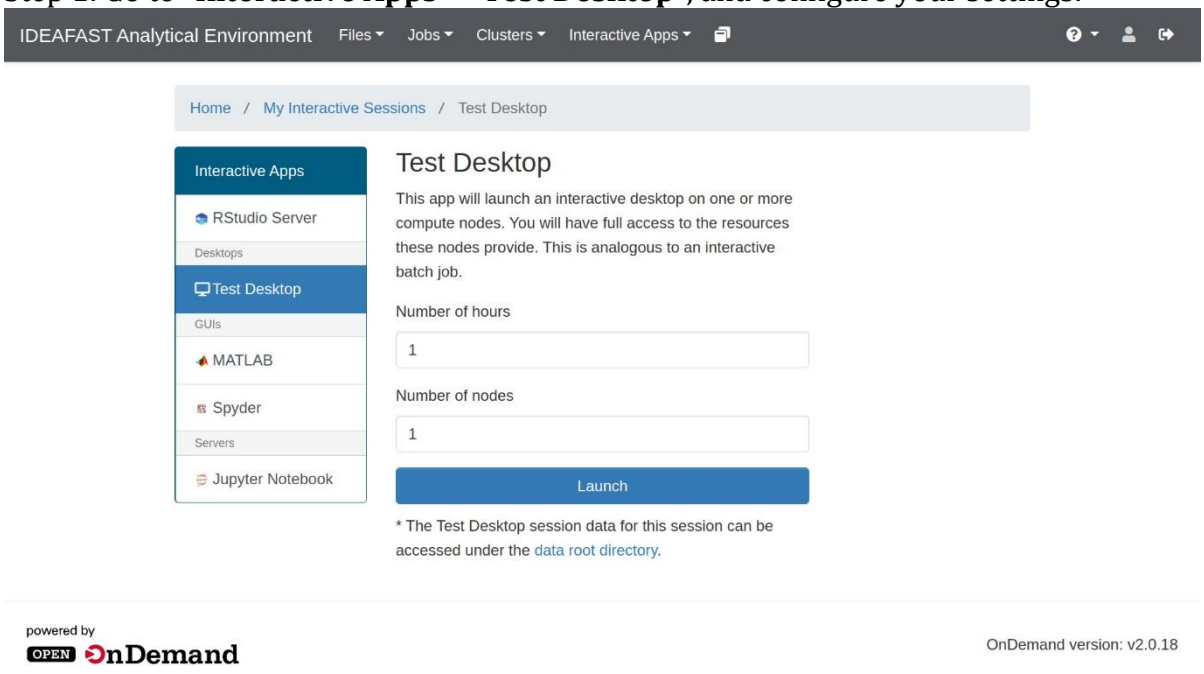


Step 2: Check the outputs based on your scripts. We use a script that simply outputs a sentence and saved in a file. Go to the folder of the scripts and you can see the outputs there. You can click the **"Files"** at the top of this page to access the folder.

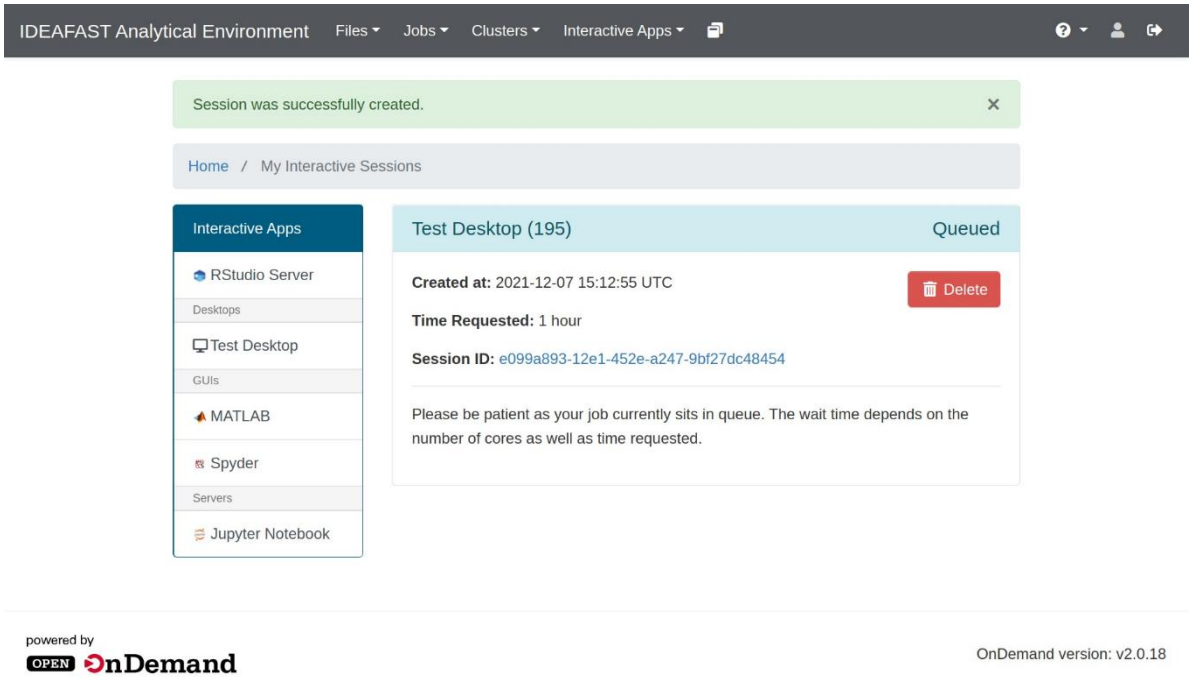


Using Desktop:

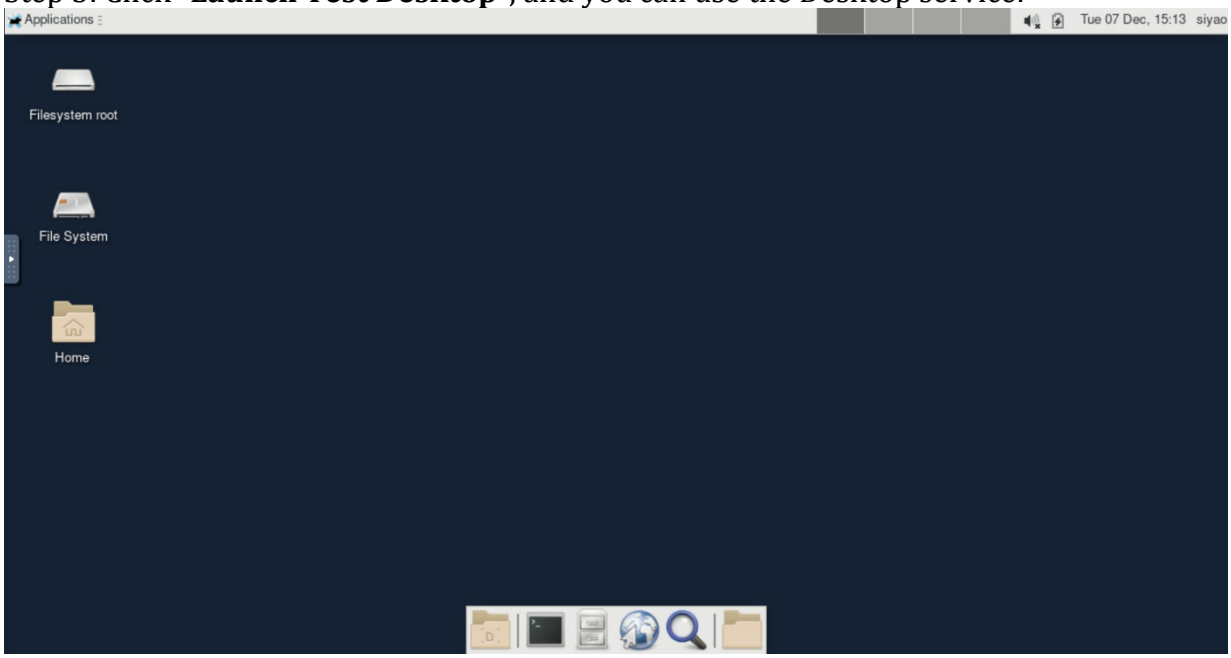
Step 1: Go to "**Interactive Apps -> Test Desktop**", and configure your settings:



Step 2: Click "**Launch desktop**" to open the Desktop service, you may need to wait for some time:

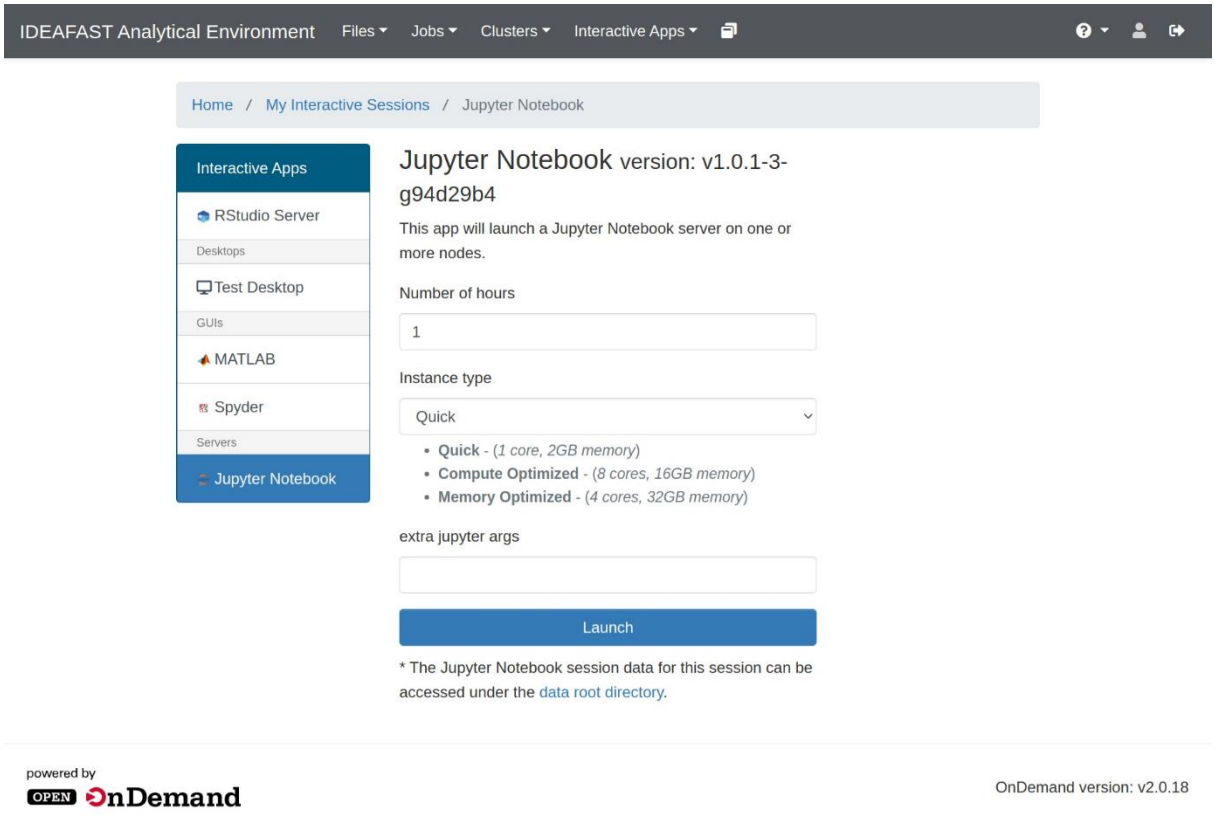


Step 3: Click "**Launch Test Desktop**", and you can use the Desktop service:

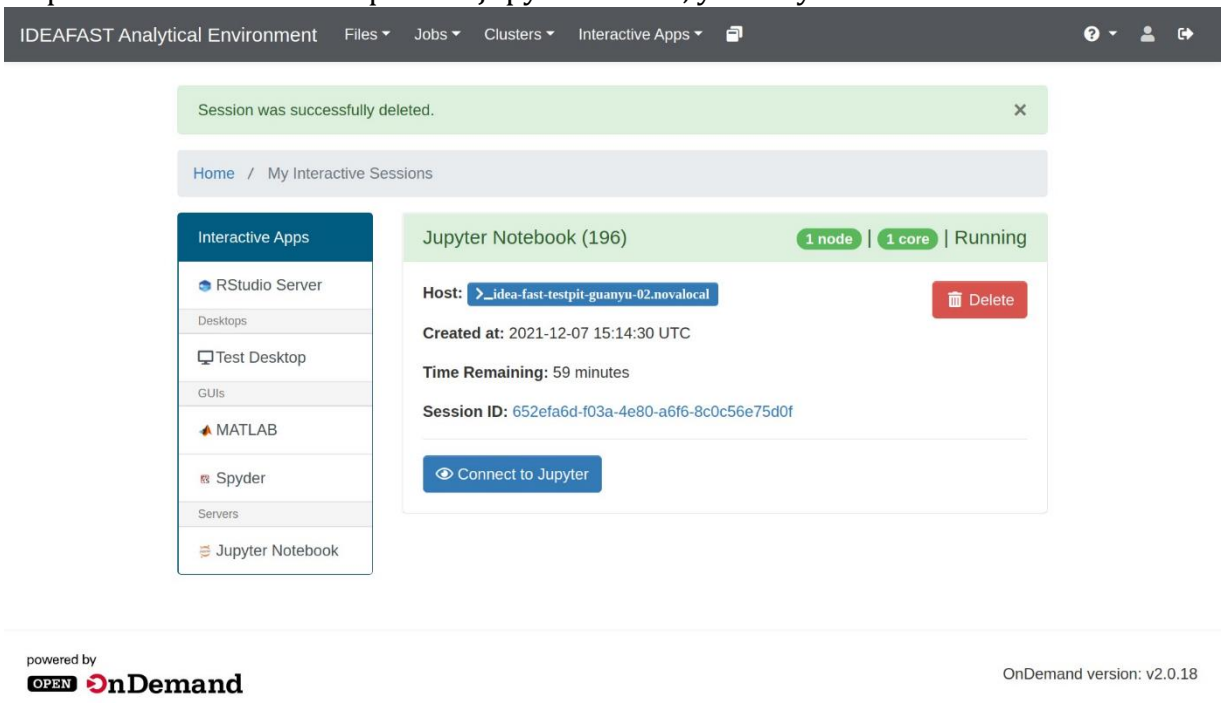


Using Jupyter Notebook:

Step 1: Go to '**Interactive Apps -> Jupyter Notebook**', and configure the settings:



Step 2: Click "**Launch**" to open the Jupyter service, you may need to wait for some time:



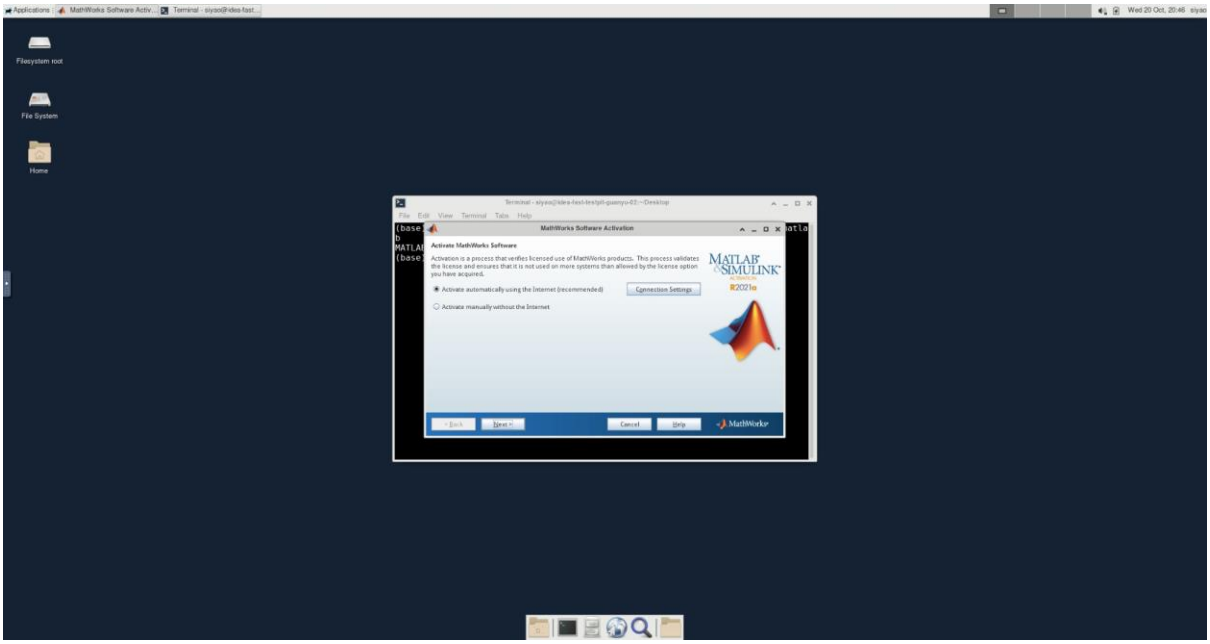
Step 3: Click "**Connect to Jupyter**", and you can use the Jupyter service:

Using Matlab:

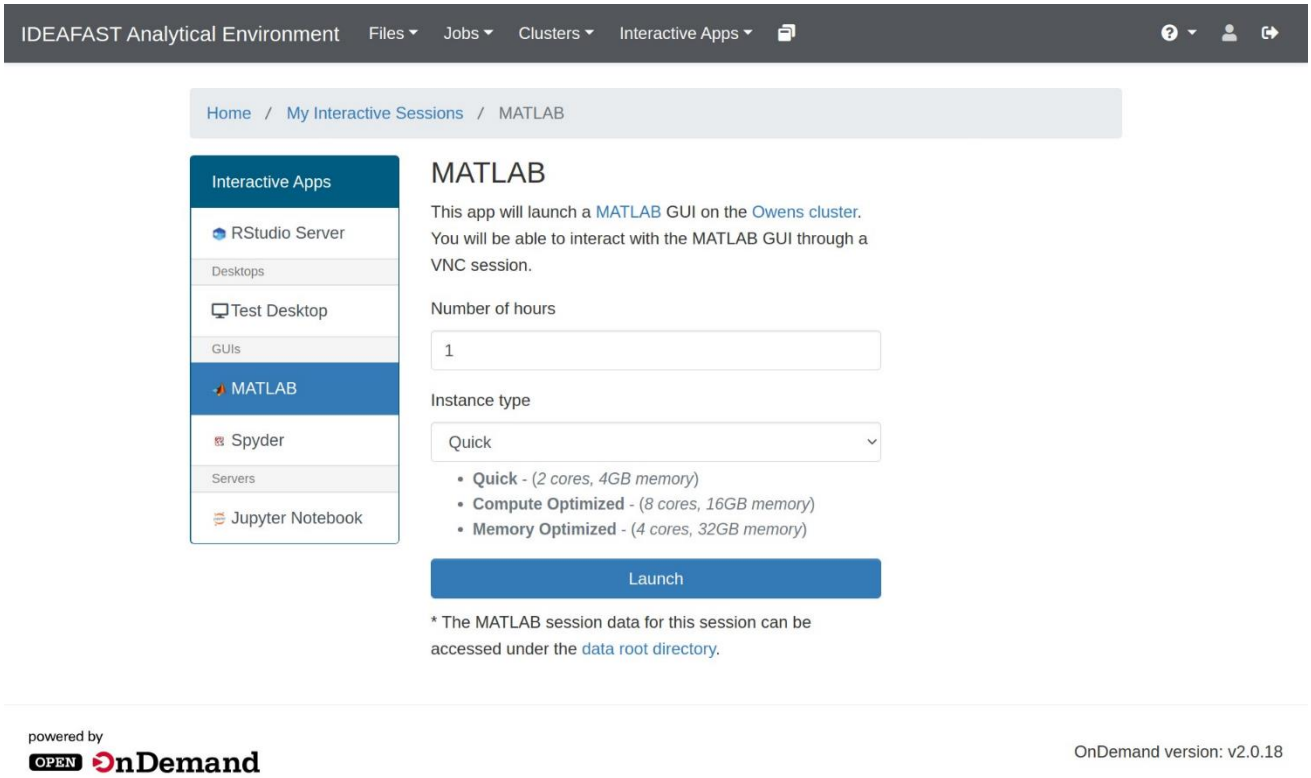
Pre-Requisites:

Matlab requires a valid license to use, which the analytical environment will not provide. You need to use your own licence. To activate the Matlab:

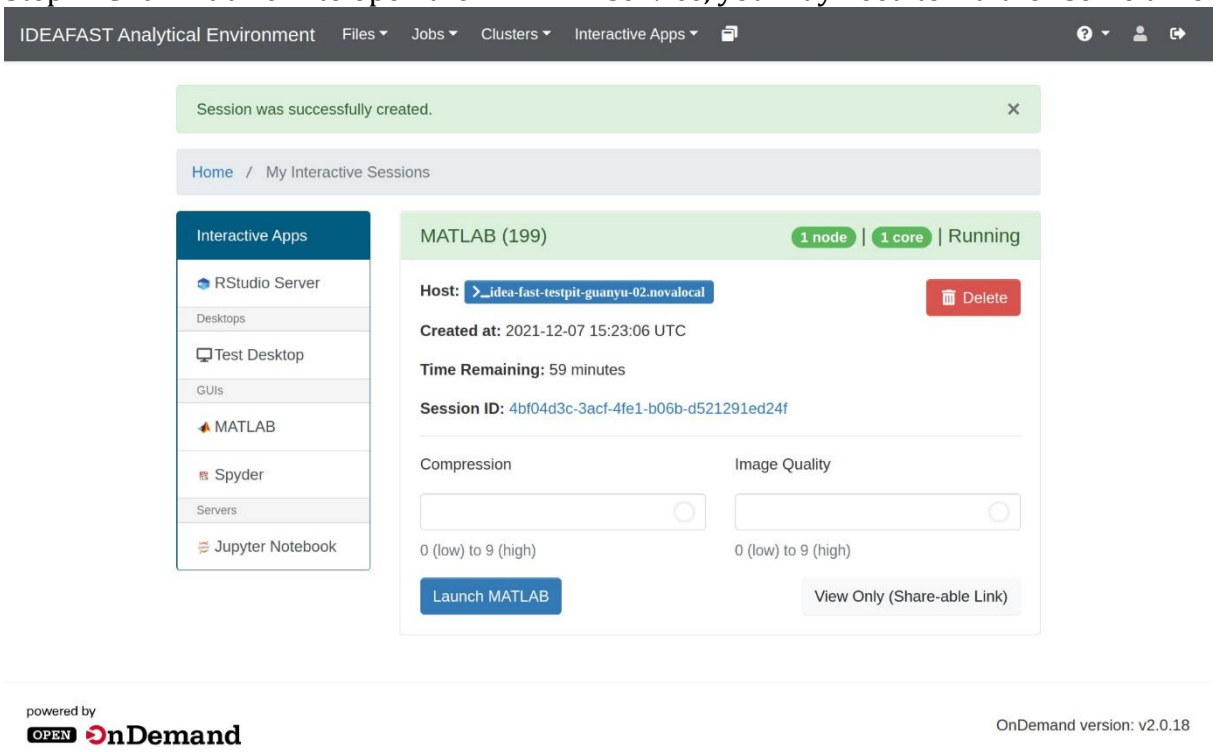
1. Open a Desktop Service
2. Open the Matlab in the terminal. Right click the "**Open Terminal Here**" to open a terminal, then open Matlab using command: */mnt/MATLAB/R2021a/bin/matlab*
3. Follow the instructions to active your matlab:



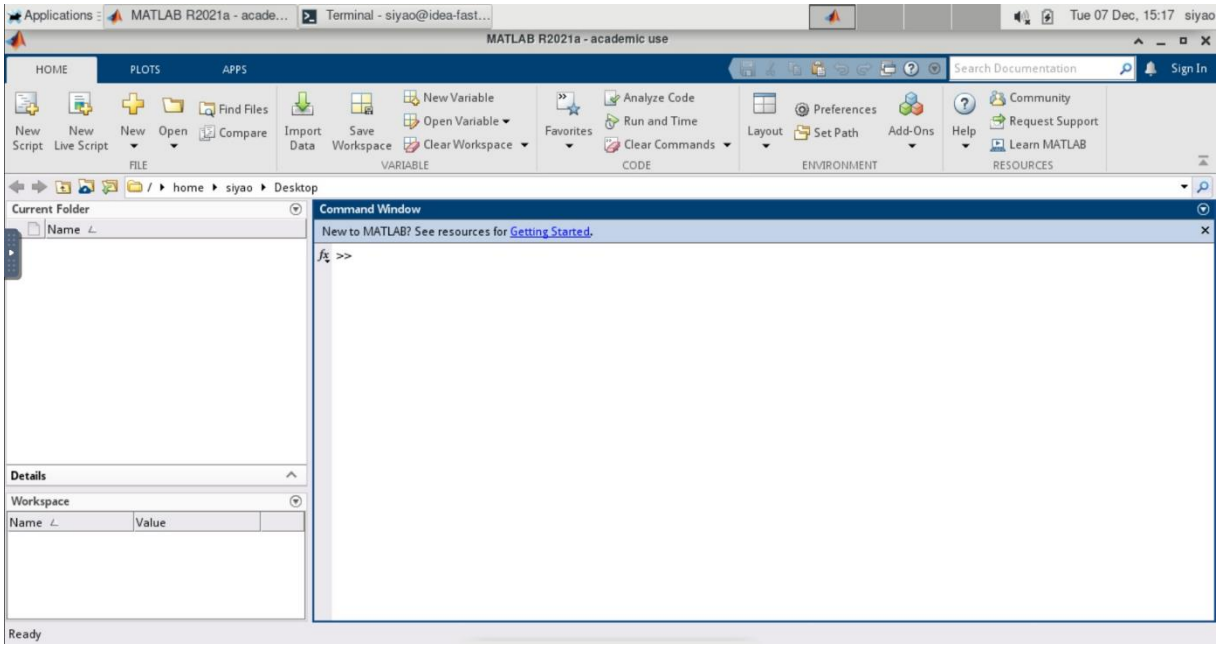
Step 1: Go to "**Interactive Apps -> MATLAB**", and configure your settings:



Step 2: Click "**Launch**" to open the MATLAB service, you may need to wait for some time:

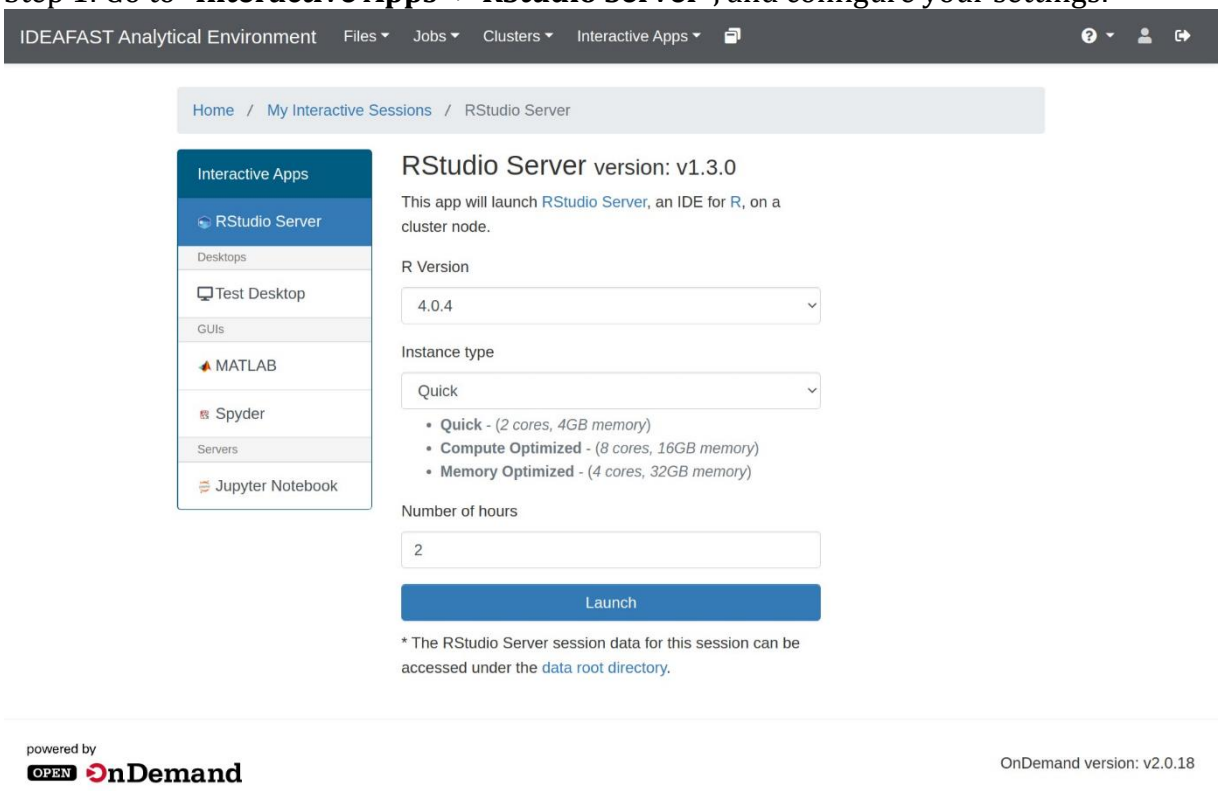


Step 3: Click "**Launch Matlab**", and you can use the Matlab service:

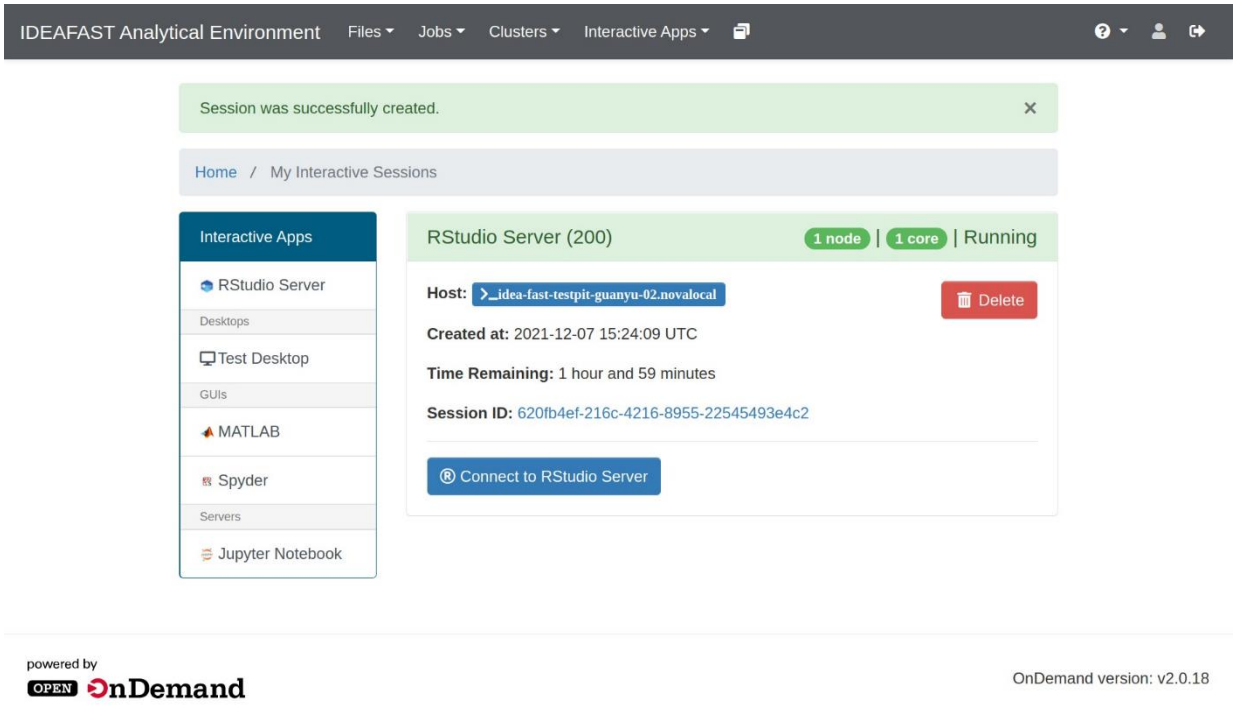


Using RStudio Server:

Step 1: Go to "**Interactive Apps -> RStudio Server**", and configure your settings:



Step 2: Click "**Launch**" to open the RStudio service, you may need to wait for some time:



Step 3: Click "**Connect to RStudio Server**", and you can use the RStudio service: