# Product description

In this project, we have built a proof-of-concept prototype hospital information system (HIS), with basic security features. The product can be accessed on \_\_\_\_\_\_\_\_\_. The following is a description of the main product:

There are 10 modules included in the prototype HIS, each should be placed on a separated machine. The 10 modules are: Front desk, EMR, LIS, Lab analyzer, RIS, Imaging modality, PACS, Doctor’s workstation, CDSS and Authentication server. With the exception of Authentication server, each module contains a script (hereinafter referred to as “the functionality script”) which facilitates the normal functionalities of the module and a script (hereinafter referred to as “the security script”) for security features. The following is a breakdown of these modules:

For all modules: Security, networking and database configurations are at the beginning of each functionality script, after the library imports. Configuration should be consistent in order to have the system function properly.

Authentication Server: This module is connected with every other module in the system. It provides an id-password based authentication process for other modules. The default id-password combinations are provided in the healthcare\_worker.db file. The passwords are stored as hash values. These id password combinations are:

|  |  |
| --- | --- |
| ID | Password |
| 10001 | bobdylon |
| 10002 | blowinginthewind |
| 20001 | johnnycash |
| 20002 | hurt |

The user can add more ID-password combinations by directly modifying the database file.

Security.py: This script contains 3 security features, namely 2 factor authentication (2FA), Digital signature and symmetric encryption. The encryption key and the secret for generating one-time passwords (OTP) are hard coded in the script for simplicity (this can be changed if needed). The script for 2FA set-up is not included in the Github repository, but uploaded directly onto myuni. Upon running the set-up script, a secret as well as a QR code will be generated and displayed. Scan the QR code with Google Authenticator and replace the 2FA secret on each security script with the newly generated secret for the 2FA feature to function. For digital signatures, the public key is renamed key\_val and private key key\_sign. The keys are included in pertinent modules, but the path to the keys still need to be reconfigured, especially for Windows machines. To switch on and off these features, change the corresponding boolean value in the Security CONFIG section at the beginning of each functionality script. So far, the encryption feature hasn’t been optimized and having inconsistent settings on encryption across different modules would lead to errors.

EMR: This module is used both as a database for patient data storage and as a communication hub for other modules. 2 tables are stored in the database file, one for patients’ clinical information, including patient id, name, date of birth, etc, the other for doctor’s consultation orders. Communication between EMR and other modules are facilitated with HL7 protocols. There is a function in the functionality script of these module which routes HL7 messages to different handlers depending on the types of messages the EMR is handling. This module should be running in order to use Front desk, Doctor’s workstation, LIS and RIS.

Front desk: This module deals with patient registration, ordering laboratory tests and radiology exams, and book doctor consultations. You can also use the front desk to view the inpatient list and remove patients from the list as well as the EMR database. There is a tkinter based GUI in this module so it should be run on a GUI-supporting system.

LIS: This module deals with laboratory test workflows. Orders initiated at the from desk will go through the EMR and be forwarded to this module, where it will be stored in a local database. You can view the list of orders and send selected orders (by right clicking on the order list) to the lab analyzer to get the result. Once the result is generated by the lab analyzer, it would be returned to the LIS and stored in another local database, which can also be viewed. A summarized version of the test result will be sent to the EMR to update the “test\_result” field of the corresponding patient in the patients database. Communication is facilitated with HL7 protocols.

Lab analyzer: This module simply takes in the orders sent by the LIS, and based on the test type, generates a random number between 20% below the lower bound and 20% above the upper bound of the normal range of the type of test as the result, then formulate a string containing the test result and send it back to the LIS.

RIS: This module receives the radiology exam orders from the EMR and stores them in a local database, as does the LIS. But instead of forwarding individual orders to Imaging Modality like the LIS does to the Lab analyzer, it passively waits for Imaging Modality to query for the order list (hereinafter referred to as the modality work list (MWL)). It does not store medical imaging results, but can retrieve the results from PACS and display them. Because of the GUI and image display features of this module, it should be placed on a GUI-supporting system. Communication between RIS and EMR is facilitated by HL7 protocols, and that between RIS and imaging modality regarding MWL retrieval, as well as RIS and PACS, is by DICOM protocols.

Imaging modality: This module contains an images directory with 4 sub directories, each containing a brain CT image. Upon launching, it actively queries the RIS for the MWL, and displays the first order on the terminal, waiting for a keyboard input of “S” to confirm and start the imaging process. Once confirmed, it randomly chooses one image from the images directory and build a DICOM file with it and the corresponding patient’s metadata such as patient id, name, etc. If the digital signature feature is on, it also embeds a signature within the DICOM file. Then it returns an HL7 message to the RIS while sending the DICOM file to PACS.

PACS: This module is used for DICOM file storage and query. After receiving DICOM files from the imaging modality, it creates a local database to store the information regarding the files, then it encrypts and stores the files in a local directory. The encrypted DICOM files are named x.dcm.enc, with x being the SOP instance UID generated during DICOM file creation on the imaging modality. The local database serves to map patient ids to SOP instance UIDs, so the DICOM files can be correctly associated with the corresponding patients.

Doctor’s workstation. This is the module representing a doctor’s office computer. After logging in, you can retrieve the doctor’s consultation orders from the EMR, and for each order, right click and view the corresponding patient’s laboratory test summary or clinical comments. You can also add clinical comments to the patients and the diagnosis field of the corresponding patient in the patients database on the EMR will be updated accordingly. You can also retrieve and view the patient’s medical images like on the RIS. For the image, you can forward them to CDSS\_AI for AI analysis.

CDSS: This is an AI server. The model weight file is needed for it to function but not included in the Github repository as it has exceeded the Github file size limit, therefore the weight file is uploaded directly via myuni submission. The weight file should be placed in the model directory of this module for it to function. This module simply takes the DICOM file forwarded by the Doctor’s workstation, extracts and, if configured so, validates the image before passing the pixel data to the model for analysis. The result is then returned to the Doctor’s workstation for display.