	OpenMRS (latest stable release: version 1.8.3)
URL	http://openmrs.org/
Description	The Open Medical Record System (OpenMRS®) is an open source medical record platform for developing countries. It is a common platform upon which medical informatics efforts can be built. The system is based on a conceptual database and can be customized for different uses. It allows implementers to design a customized medical records system with little or no programming skills. OpenMRS features include a Central concept dictionary, Modular architecture and Standards support.
Licensing	OpenMRS is distributed under the OpenMRS Public License 1.1. This license is based on the Mozilla public license version 1.1, but contains certain terms and conditions that differ. (See section 6.3 of https://wiki.openmrs.org/display/RES/OpenMRS+Public+License+1.1 for these differences)
Cost	Free
Support	OpenMRS volunteers, core developers and implementers maintain developer and implementer mailing lists. OpenMRS has its own answer support system. The core team also hosts weekly calls to share knowledge. However, free support is not guaranteed for implementations. Alternatively, they (implementers) may open tickets and ask for guidance.
Community	A large and active community of volunteers, professionals and academics. Also maintains mailing lists and a wiki.
Community URL	http://openmrs.org/

Source code availability	Source code is freely available
Active Development	Yes, by OpenMRS core developers employed by the Regenstrief institute and PIH, Boston. Also supported by a large number of volunteer developers.
Language Support	Documentation is available only in the English language. However the product provides Localization / internationalization (currently supports English US, English UK, Italian, Spanish, French and Portuguese). OpenMRS also supports the possibility to extend to other languages with full UTF-8 support.
Operating Systems	OpenMRS is cross-platform, and written in Java. It uses a Spring/Hibernate/maven stack. There is no specific preferred OS. Developers use Windows, Linux and Mac OSX.
Database Systems	Version 1.8.2 only supports MySQL. However version 1.9 (yet to be released) includes changes to support Postgresql and Sql Server.
Interoperability	OpenMRS can be accessed via both REST and SOAP web services. The recommended (and only supported method) is via the Webservices.rest Module. The PIX/PDQ module supports basic PIX/PDQ transactions. OpenMRS also provides a SMART container for SMART applications.
Current Installations	OpenMRS implementations exist in South Africa, Kenya, Rwanda, Lesotho, Zimbabwe, Mozambique, Uganda, Tanzania, Haiti, India, China, United States, Pakistan, Philippines, Sri Lanka, Indonesia and many other countries. A detailed list of implementation sites, contact persons and approximate sizes can be seen at (http://openmrs.org/about/locations/)
Performance Metrics	

Matching algorithms	The OpenMRS Patient Matching module is built on the Felligi-Sunter probabilistic matching algorithm.

Requirements	**Scope**	**Priority**	**Fully Complies**	**Partially Complies**	**Does not Comply**	**Notes + Further Information**
1 Shared Health Records						
Functional Requirements						
1.1 Information Request						
1.1.1 System must retrieve an appropriate record in response to a request.	Y	1		✓		
1.1.2 System must validate the request location	N					//Does not apply, removed. To be handled by the Interoperability layer.
1.1.2 System must respond to time based and cohort queries	Y- but I believe we agreed this is part of the messaging NOT the SHR	3		✓		We will be supporting the queries defined here. We will not be supporting Cohort Based queries for the Phase 1 Project Implementation. An Example Use Case of a cohort based query is : From the POC perspective, if we have a scheduling solution in place for ANC visits, we could make batch requests to pull down the latest encounter data from the SHR some time before the actual visit.
1.1.3 System must identify newly updated information	Y	1	✓			New information will be returned based on messaging based requests
1.1.4 System must have security rules to validate data request	N- we also decided there is NO UI so this is not applicable	1	✓			From the interoperability Layer. For now, we'll be supporting Basic Authentication or similar between the interoperability layer and the SHR.

1.1.5 System must enforce different security requirements on different data	N- see above	2			✓	OpenMRS persists data as Observations. We cannot enforce restrictions for only a given set of data. Any user who has permissions to view observations may view all observations.
1.1.6 System must accommodate externally defined roles, which are equivalent to those in the Provider Registry.	N	2	✓			Basic Role Based Security will be supported which OpenMRS currently provides.
1.1.7 System must maintain an audit log of all information requests whether they are completed or not	Y	2		✓		The Access Logging module (requires 1.4.0) will allow access logging for Patients and Encounters. However, there is no provision for uncompleted requests. We will just log all communications with the interoperbaility layer.
1.1.8 System must provide an acknowledgment for each information request. The acknowledgement may be either a success or error message	Y	1			✓	Refer to Transaction specifications document for the responses we have identified.
1.1.9 System must identify when a subset of the available information is returned due to insufficient credentials etc.	N- see above	3			✓	OpenMRS favors to reject the entire request in such an event*. Therefore. No such identification is made.
1.1.10 System must retrieve information in specified aggregates or domain specific subsets	N- not needed to support our use case	3	✓			For the Phase 1 Implementation, we will only be supporting the queries defined in the transaction specification which will be initiated from the OpenMRS POC. Any additional reporting is out of scope.
1.1.11 System must export either the full set of data for a particular patient, or a predefined subset based on user triggered predefined subsets.	N	3	✓			We will support all queries defined in the transaction specification document.
1.2 Information Storage						
1.2.1 System must validate that patient, provider and location is valid.	N					//Does not apply, removed. To be handled by the interoperability layer

1.2.1 System must validate the content is using the appropriate vocabulary.	N					//Does not apply, removed. To be handled by the TS through the interoperability layer.
1.2.1 System must be able to perform saves in an asynchronous fashion. (via usb or a queue).	Yes!	2		✓		The USB use case still needs to be defined further.
1.2.2 System must retrieve both persistent (e.g. allergies) person and encounter data (based on security restrictions)	Y- this will have a lot to do with our data model	2	✓			Again, we will support the defined queries in the transaction specification.
1.2.3 Successful transactions must be acknowledged to originating system	Y	1			✓	Refer to Transaction specifications document for the responses we have identified.
1.2.5 System must store unsuccessful transaction data along with reasons for failure	y- but could be in an error queue	1		✓		The Access Log module is the closest contender to support this functionality. However neither this module records unsuccessful transaction data.
1.2.6 In the event of an unsuccessful transaction, the originating system must be notified	Y	1			✓	
1.2.7 System must be able to correct/address any unsuccessful transactions.	y- but this will likely be MANUAL at first	1	✓			
1.2.8 System must be able to aggregate several episodes together (care composition)	y- again goes to data model	3	✓			Although we can support this, we do not have specific use cases for our Phase 1 Implementation, besides returning multiple encounters for our messaging requests.
1.2.9 System must be able to create trigger events for clinical decision support or alerts	Y- likely only 1 or 2- I will tell you what they are specifically soon	1	✓			OpenMRS uses Arden and the DSS module to provide clinical decision support.

1.2.10 System must be able to support all data types	y- although lets discuss photos- I think out of scope- but we need to make sure we don't build something that does not let us add this later	1	✓			OpenMRS supports all data types from numbers to Strings to concepts and text / image files.
1.2.11 System must process asynchronous requests in the order that they were created, not the order which they were received.	y- part of the interop layer I believe					We agreed at the project meeting to process messages in the SHR based on the Encounter Date and not on the Message Time-Stamp in which it was received.
1.3 Interface / Communication						
1.3.1 System will not use the end user interface	Y		✓			
1.3.2 System data must be available for reporting	Y	1	✓			Can be done using the Reporting module, Patient Summary module or the Clinical summary module
1.3 3 System must record and version all data updates	Y	2	✓			OpenMRS lets users log data changes using logs or the Access Log module. In the event of updating data, OpenMRS retires the old object and introduces a new one with the modified data, thus 'versioning' the data.
1.3.4 System must record the author of each data change	Y	1	✓			
1.3.5 In the event of a data change, the system must record a user comment on the reason for the change	Y	1	✓			
1.4 Business Rules						
1.4.1 System must define and implement business rules that are	y- to be	1	✓			As stated above, this can be done using Arden

triggered by the entry of specific data	defined 1-2 specific rules					and the DSS module
1.5 Non-Functional Requirements						
1.5.1 System must have a data recovery and back-up solution	Y	1	✓			In addition to the traditional MySQL database backup, OpenMRS also provides the Database backup module, which allows users to include or exclude selected data from the backup process.
1.6.2 System must allow for real time updates	Y	1	✓			
1.6.3 System must provide sufficient support documentation in English	Y	1	✓			Extensive wiki documentation available
1.6.4 System must be easily scalable to (a minimum) of double the current Patient count in Rwanda	Y	1	✓			
1.6.5 System must provide user restricted security to the SHR configuration	Y	1	✓			
1.6.6 System must ensure maximum possible uptime	Y	1	✓			
1.6.7 System must be able to support an x number of concurrent users	NA- this is immaterial as the user only interacts with the POC system	1	✓			//In the worst case scenario, how much concurrent users can we expect?
1.6.8 SHR database must support database change management so that database modifications can be made	Y		✓			OpenMRS uses liquibase to support database change management

Priority

1. High Priority – must be done as part of RHEA project
2. Medium
3. Low