Software Project Assessment Guideline for BTech Students

You have organised yourselves into groups and started working on your end of year project. The First and most important thing is that you shall need to present a working project at the end of next Month.

In addition to a working Project, the your software will be assessed based on sustainability, maintainability, and usability using a criteria-based assessment which gives a measurement of quality in a number of areas. These areas are derived from ISO/IEC 9126-1 Software engineering — Product quality 1 and include usability, sustainability and maintainability.
The assessment involves checking whether your software, and the project that develops it, conforms to various characteristics or exhibits various qualities that are expected of sustainable software.

Below is the list metrics that will be used. Not ALL of the ISO/IEC 9126-1 metrics are included.

Please note that this is a lot of work and it is important that you make use of your group dynamics to allocate resources to the task of complying with this.

Metric	Details	Notes
Usability	Understandability	
		How straightforward is it to understand:
		What the software does and its purpose?
		The intended market and users of the software
		The software’s basic functions?
		The software’s advanced function
		High-level description of what/who the software is for is available.
		High-level description of what the software does is available.
		High-level description of how the software works is available.
		Design rationale is available – why it does it the way it does.
		Architectural overview, with diagrams is available.
		Descriptions of intended use cases are available.
		Case studies of use are available.
	Documentation	
		Looking at the user documentation, what is its
		--->Quality?
		--->Completeness?
		--->Accuracy?
		--->Appropriateness?
		--->Clarity?
		Provides a high-level overview of the software.
		Partitioned into sections for users, user-developers and developers (depending on the software)
		States assumed background and expertise of the reader, for each class of user.
		Lists resources for further information.
		Further information is suitable for the level of the reader, for each class of user
		Is task-oriented.
		Consists of clear, step-by-step instructions.
		Gives examples of what the user can see at each step e.g. screen shots or command-line excerpts.
		For problems and error messages, the symptoms and step-by-step solutions are provided.
		States command names and syntax, says what menus to use, lists parameters and error messages exactly as they appear or should be typed.
		Uses teletype-style fonts for command-line inputs and outputs, source code fragments, function names, class names etc.
		English language descriptions of commands or errors are provided but only to complement the above.
		Plain-text files (e.g. READMEs) use indentation and underlining (e.g. === and ---) to structure the text.
		Plain-text files (e.g. READMEs) do not use TAB characters to indent the text.
		API documentation e.g. JavaDoc or Doxygen, documents APIs completely e.g. configuration files, property names etc.
		Is held under version control alongside the code. Is on the project web site (Use Wiki Page).
		Documentation on the project web site (Wiki Page) makes it clear what version of the software the documentation applies to.

Usability	Buildability	
		How straightforward is it to Meet the pre-requisites for building the software on a build platform?
		How straightforward is it to Build the software on a build platform?
		Web site (Wiki Page) has instructions for building the software.
		Source distributions have instructions for building the software
		An automated build (e.g. Make, ANT, custom solution) is used to build the software.
		Web site lists all third-party dependencies that are not bundled, along with web addresses, suitable versions, licences and whether these are mandatory or optional.
		Source distributions list all third-party dependencies that are not bundled, along with web addresses, suitable versions, licences and whether these are mandatory or optional.
		Dependency management is used to automatically download dependencies (e.g. ANT,Ivy, Maven or custom solution).
		All mandatory third-party dependencies are currently available.
		All optional third-party dependencies are currently available.
		Tests are provided to verify the build has succeeded.
	Installability	
		How straightforward is it to Meet the pre-requisites for the software on a target platform?
		How straightforward is it to Install the software onto a target platform?
		How straightforward is it to Configure the software following installation for use?
		How straightforward is it to Verify the installation for use?
		Web site has instructions for installing the software.
		Binary distributions have instructions for installing the software.
		Web site lists all third-party dependencies that are not bundled, along with web addresses, suitable versions, licences and whether these are mandatory or optional.
		Binary distributions list all third-party dependencies that are not bundled, along with web addresses, suitable versions, licences and whether these are mandatory or optional.
		Dependency management is used to automatically download dependencies (e.g. ANT, Ivy, Maven or custom solution).
		Tests are provided to verify the install has succeeded.
		When an archive (e.g. TAR.GZ or ZIP) is unpacked, it creates a single directory with the files within. It does not spread its contents all over the current directory.
		When software is installed, its contents are organised into sub-directories (e.g. docs for documentation, libs for dependent libraries) as appropriate.
		All source and binary distributions contain a README.TXT with project name, web site, how/where to get help, version, date, licence and copyright (or where to find this information), location of entry point into user doc.
		All GUIs contain a Help menu with commands to see the project name, web site, how/where to get help, version, date, licence and copyright (or where to find this information), location of entry point into user doc.
		All other content distributed as an archive contains a README.TXT with project name, web site, nature, how /where to get help, date.
		Installers allow user to select where to install software.
		Uninstallers uninstall every file or warns user of any files that were not removed and where these are

Sustainability and maintainability	Identity	
		To what extent is the identity of the project/software clear and unique both within its application domain and generally?
		Project/software has its own project site.
		Project/software has a logo.
		Project/software has a distinct name within its application area.
	Copyright	
		To what extent is it clear who wrote the software and owns its copyright?
		Project site states copyright.
		Project site states who developed/develops the software, funders etc.
		If there are multiple web sites then these all state exactly the same copyright, licencing and authorship.
		Each source code file has a copyright statement.
		If supported by the language, each source code file has a copyright statement embedded within a constant.
		Each source code file has a licence header.
	Licencing	
		Has an appropriate licence been adopted?
		Project Web site states licence.
		Software (source and binaries) has a licence.
	Governance	
		To what extent does the project make its management, or how its software development is managed, transparent?
		Project has defined a governance policy.
		Governance policy is publicly available.
	Testability	
		How straightforward is it to test the software to verify modifications?
		Project has unit tests.
		Project has integration tests.
		For GUIs, project uses automated GUI test frameworks.
		Project has scripts for testing scenarios that have not been automated (e.g. for testing GUIs).
		Project recommends tools to check conformance to coding standards.
		Project has automated tests conformance to coding standards.
		Continuous integration is supported – tests are automatically run whenever the source code changes
	Changeability	
		How straightforward is it to modify the software to Address issues?
		How straightforward is it to modify the software to Modify functionality?
		How straightforward is it to modify the software to Add new functionality?
		Project has defined a contributions policy.
		Contributions policy is publicly available.
		Project has defined a stability/deprecation policy for components, APIs etc.
		Stability/deprecation policy is publicly available.
		Releases document deprecated components/APIs in that release.
		Releases document removed/changedcomponents/APIs in that release.