DUY TÂN UNIVERSITY

International School

CAPSTONE PROJECT 2

CMU-SE-451 / CMU-IS-451 / CMU-CS-451

PROJECT PLAN

Version 2.0

Date: 26 - Mar - 2021

EXPERT-DRIVEN SMART DASHBOARD APPLICATION

Submitted by

Vo Van Hoa
Pham Van Tin
Ky Huu Dong
Tran Thi Thanh Kieu

Approved by

Capstone Project 2 - Mentor:

Name	Signature	Date
Binh, Thanh Nguyen		26 - Mar - 2021

PROJECT INFORMATION			
Project Acronym	EDSDA		
Project Title	Expert-Driven Smart Dashboard Application		
Project Web URL	https://sda-research.ml/		
Start Date	01 - Mar - 2021		
End Date:	01 - Jun - 2021		
Lead Institution	International School, Duy Tan University		
Project Mentor	Ph.D Binh, Thanh Nguyen		
Scrum Master	Hoa, Vo	hoavo.dng@gmail.com	0935.193.182
Team Members	Tin, Pham Van	tinphamvan123@gmail.com	0932.535.175
	Dong, Ky Huu	kyhuudong@gmail.com	0898.246.980
	Kieu, Tran Thi Thanh	thanhkieutran391@gmail.com	0358.583.251

DOCUMENT INFORMATION			
Document Title	Project Plan		
Author(s)	Team C2SE.06		
Role	[EDSDA] ProjectPlan_v2.0		
Date	01 - Mar - 2021	Filename	[EDSDA] 002 Project Plan
URL	https://github.com/sdateamdtu2020/SDA-v2.0		
Access	Project and CMU Program		

REVISION HISTORY

Version	Person(s)	Date	Description
Draft	Hoa, Vo	01 - Mar - 2021	Initiate document
2.0	All members	14 - Mar- 2021	Finish content of the document
2.1	All members	25 - May - 2021	Update content, fix typo

TABLE OF CONTENTS

1. PROJECT OVERVIEW

1.1. PROJECT DESCRIPTION

Project code	C2SE.06	**Contract type**	Scientific Research
Customer	International School, Duy Tan University	**End-User**	Environmental Experts, Users
Project Type	External	**Scrum master**	Hoa, Vo Van
Project Category	Development & Maintenance	**Business domain**	Data Visualization, Environment,...
Application type	Web Application		

1.2. SCOPE AND PURPOSE

1.2.1. SCOPE

The backbone of the project :

- Import data sources.
- Auto ETL
- Auto RDF Generation.
- Auto linking data.
- Drag data cubes from the cubes list.
- Drop data cubes onto the main content board.
- Connect the data cubes that have the relation between them.
- Merge the cubes to create a new data cube based on connected data cubes.
- Visualize the data cube as a line chart, a pie chart, a column chart, or a geographical map.

Language:

- English

Duration: 13 weeks.

1.2.2. PURPOSE

Expert-Driven Smart Dashboard - means a dashboard that is convenient for users to analyze and review data. It will include several datasets about real-time information of the environment, measurement data of air pollutants... EDSDA will connect and analyze data from multiple sources in ways you've never imagined. Then reveal the insights you've been missing...in just a matter of minutes.

With smart suggestions and an intuitive visual interface, SDA makes it easy for any user to combine data and discover hidden insights in one place...without the usual scripting, coding, and IT hand-holding. With this dashboard, individuals or any subjects can take advantage of environmental data to be able to decide the best relevant policies.

Not only for the environmental expert, but EDSDA also helps the user directly interact with their own data source by importing, automatically cleaning their data, and enabling the user to do everything with their own data in the easiest way.

1.3. ASSUMPTIONS AND CONSTRAINTS

No	Description	Note
Assumptions		
1	Auto ETL and RDF generation may not be able to finish on time	Scope
2	The system will be uploaded to the web app for a public user acceptance test	External Interfaces
3	The system will be tested with pre-prepared standard data sources	Data Sources
Constraints		
1	EDSDA must be done and delivered by June 30th according to the customer demonstration deadline	Schedule
2	The project must conform to the system architecture specification and end-user requirements	Consistency
3	The project must comply with information security rules	Security

CONSTRAINTS	CONSTRAINTS DESCRIPTION	GUIDELINES FOR ACCEPTANCE
Economic	In terms of cost, since it is a dashboard with not too many screens, the main issue lies in the cost of researching, implementing a fully automated backend system and frontend with smart methods. There is also the cost of renting a server and deploying a server that should be considered carefully	Design cost: Must be around $100. Production cost: Should be under $4000. Maintenance cost: Should be around $500. Operation cost: Should be under $1000
Ethical	Because this software analyzes data with most of the data being crawled, so pay attention to copyright and data security.	All crawl data sources must be open data and be allowed for public use.
Social and Global	The software is targeted to not only environmental experts but also data analysts in many other fields, so widespread popularity is very important.	The product needs to be developed in an optimal and user-friendly way to reach a wide range of users
Sustainability	Need to maintain the continuous operation of the system, so as not to affect the analysis of people	Development and maintenance work must be ensured to take place continuously, when issues are reported, it is necessary to focus on maintenance immediately. Server operation also needs to be ensured not to be interrupted.

1.4. PROJECT OBJECTIVES

1.4.1. Standard Objectives

Metrics	Unit	Committed	Note
Start Date	dd-mmm-yy	01/Mar/21	
End Date	dd-mmm-yy	01/Jun/21	
Duration	elapsed days	91 days	
Maximum Team Size	Person	4	
Billable Effort	Person-day	$10/day	
Calendar effort	Person-day	$10/day	
Effort Usage	Person-day	364	

1.4.2. Specific Objectives

- Build a dashboard with drag-and-drop intelligence yet user-friendly.
- Implement a data warehouse that has a fully automated ETL method.
- Build a data import system and automatically clean data to help users automatically clean and interact with their own data source.
- The engine to automatically generate multidimensional RDF data cubes from the data warehouse must be built.
- The GraphDB database management system must operate continuously and allow data access up to 10000 rows.

1.4.3. Quality Attributes

ID	QA01
Quality Attributes	Performance
Stimulus	Uses the statistics merge operator for merging multiple data cubes.
Source(s) of stimulus	User
Artifacts	System
Environment	Normal mode
System response	The SDA returns a new data cubes as fast as possible
Response measure(s)	Within 3 seconds

Table 3.3.1: Quality Attributes: Performance

ID	QA02
Quality Attributes	Availability
Stimulus	The power is off while server is running
Source(s) of stimulus	Power
Artifact	During peak usage load
Environment	Hardware and software
System response	The system will use the cloud server to save the work so we don't need to worries about power incident occurred
Response measure(s)	All work always can be saved

Table 3.3.2: Quality Attributes: Availability

ID	QA03
Quality Attributes	Availability
Stimulus	Can't get specific data cubes when selecting it from the widget
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	The system will log the fault immediately
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Availability

ID	QA03
Quality Attributes	Portability and compatibility
Stimulus	Open the browser and access http://sda-research.ml/
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	The system can run on any web browser.
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Portability and compatibility

ID	QA03
Quality Attributes	Security
Stimulus	User access to EDSDA and import a file
Source(s) of stimulus	User
Artifact	System
Environment	Normal mode
System response	The system will not track any individual information about the user
Response measure(s)	Within immediately

Table 3.3.2: Quality Attributes: Availability

1.5. CRITICAL DEPENDENCIES

No	Dependency	Expected delivery date	Note
1	Cloud Server	May 15th	put all the necessary modules on the server cloud to ensure the continuity of the system
2	GraphDB	April 30th	Store and retrieve linked data warehouses as RDF Data Cubes
3	Sample Data Sources	April 20th	Used to test import methods and automate data source ETL
4	Open data sources	April 10th	Exploiting open data sources of topics to create a stock of available data

1.6. PROJECT RISK

This part of the document contains several risks that could happen to the development team in the future. It also includes probability, severity, and mitigation strategy for each risk.

Risk	Definition	Probability	Severity	Mitigation Strategy
Lack of coding experiences	No one in team member work with Python, React, Express, Data Warehouses, Data Cubes, Crawl data	3	2	Each team member has to learn and help the other to learn quickly.
Source Code conflict	Problems while merging code between members to master branch	3	3	Each team member must resolve conflicts by using git merge CLI before merging to the master branch.
Member conflict	Team member maybe conflict with each other while discussing	3	2	Team building, playing board games to get everyone together.
Less equipment	No machine or hosting for deploying the server.	1	2	Try free hosting for deployment.
Time management	Every member has to go to work or school.	3	3	Overtime
Language barrier	Most of the documents the are in English, sometimes it hard to understand clearly the articles and the information	3	3	Improve the individual English skills while doing the project. Asking the mentor technology for specific

Probability		Severity	
1	Rarely happened.	0	Low damaged
2	Sometime happened	1	Medium damaged
3	Usually happened	2	Serious damaged

2. PROJECT DEVELOPMENT APPROACH

2.1. PROJECT PROCESS

2.1.1. Reasons for selecting

- Our team has 4 people
- The project will be continuously horizontally scaled up.
- There is only a short amount of time to finish the project.

So based on those constraints, we decided to choose SCRUM as the project lifecycle.

2.1.2. Agile Methodology

PRINCIPLE AND DIFFERENT STAGES

The SCRUM methodology relies on the incremental development of a software application while maintaining a completely transparent list of upgrade or correction demands to be implemented (backlog). It involves frequent deliveries, usually every four weeks, and the client receives a perfectly operational application that includes more and more features every time. This is why the method relies on iterative developments at a constant rhythm of 2-4 weeks. Upgrades can therefore be more easily integrated than when using a V-cycle.

This method requires four types of meetings:

- ➢ Daily meetings: the entire team meets for approximately 15 minutes every day in order to answer the following three questions, usually while standing: what did I do yesterday? What am I going to do today? Is there a cumbersome impediment today?
- ➢ Planning meetings: the entire team gathers to decide on the features that will make up the following sprint
- ➢ Work review meetings: during this meeting, every member presents what he has done during the sprint. They organize a demonstration of the new features or a presentation of the architecture. This is an informal meeting lasting for approximately 2 hours which is attended by the entire team.
- ➢ Retrospective meetings: at the end of each sprint, the team analyzes both successful and unsuccessful elements of their activity. During this meeting lasting between 15 and 30 minutes where everyone is invited and speaks on their own behalf, a vote of confidence is organized in order to decide on the improvements to be made.

The advantage of this method consists in reducing the documentation to the minimum in order to gain productivity. The idea is to write only the minimum documentation

which allows to save the history of the decisions taken on the project and to easily perform interventions on the software when it goes into the maintenance phase.

AGILE - SCRUM ORGANISATION

The SCRUM methodology involves the following three main players:

- Product owner: In most projects, the product owner is the leader of the client's project team. He is the one who will define and prioritize the product features and choose the date and content of each sprint based on values (workloads) that the team communicates to him.
- Scrum Master: He is a genuine facilitator on the project as he makes sure that everyone works at their full potential by eliminating impediments and protecting the team from external interference. Moreover, he pays particular attention to the respect of the different SCRUM phases.
- Team: A team is typically made up of 4-10 people and groups together all the IT specialists who are necessary on a project, i.e. an architect, a designer, a developer, a tester, etc. The team is self-organizing and remains unchanged during an entire sprint.

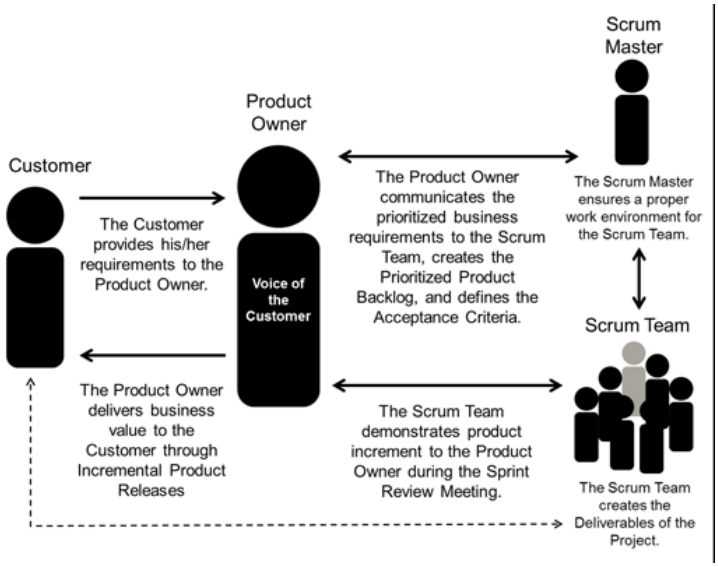

Figure 1: Scrum team members

AGILE - SCRUM ADVANTAGES

Scrum differs from other development methods through its advantages which turn it into a pragmatic response to product owners' current needs:

Iterative and incremental method: this allows to avoid the "tunnel effect", i.e. the fact of seeing the result only at the final delivery, and nothing or almost nothing during the entire development phase, which is so frequent with V-cycle developments.

Maximum adaptability for product and application development: the sequential composition of the sprint content allows the addition of a modification or a feature that was not initially planned. This is precisely what renders this method "agile".

- Participatory method: every team member is asked to express his opinions and can contribute to all the decisions taken on the project. He is therefore more involved and motivated.
- Enhancing communication: by working in the same development room or being connected through different communication means, the team can easily communicate and exchange opinions on the impediments in order to eliminate them as early as possible.
- Maximizing cooperation: daily communication between the client and the team enables them to collaborate more closely.
- Increasing productivity: as it removes certain "constraints" of the classical methods, such as documentation or exaggerated formalization, SCRUM allows for increased team productivity. By adding to this the qualification of each module which allows determining an estimation, everyone can compare their performance to the average team productivity.

2.1.3. Scrum Process

Scrum is an agile method, so it follows the principles of Agile Manifesto (see also Agile Manifesto). In addition, Scrum operates on three core values, also known as Scrip Scripps, including Scrutiny, Inspection, and Adaptation.

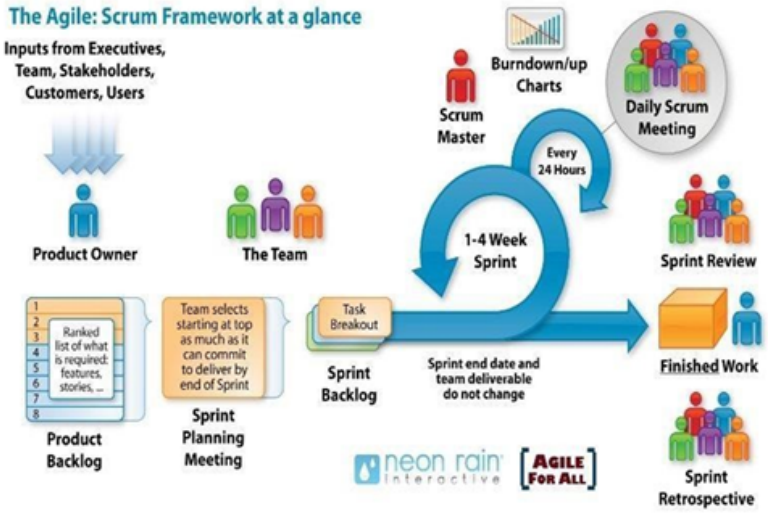

Figure 2: Scrum team members

Based on the empirical process control theory, Scrum uses iterative and incremental algorithms to optimize efficiency and control risk. Scrum is simple, easy to learn, and has wide applicability. To be able to use Scrum, we need to understand and apply the elements that makeup Scrum include the core values (also known as the "three legs", or the three pillars of Scrum), roles, Events, and Scrum-specific artifacts.

2.2. REQUIREMENT CHANGE MANAGEMENT

We use Ince's Change Process Model to handle the required changes. Ince's model focuses on how software configuration management relates to software

change management. This model has two main sources of change requests, i.e. customer and development team as shown in Figure 1. In order for the change process to be initiated, a change request must be initiated in a software project. All such change

requests are recorded in a change request note. The change control board then considers the suggested change. The change control board can reject the change (the

change will not take place), batch the change (the change will take place but not immediately), or accept the change (the change is to be implemented at the earliest possible time). If the request for the change is successful, a change authorization note

must be filled. After this, the change can be implemented and a system's documentation is modified. After implementation, the change is validated. Validation and test records are then produced to document the changes that have taken place. Finally, the configuration records are updated and the staff is informed about the new changes.

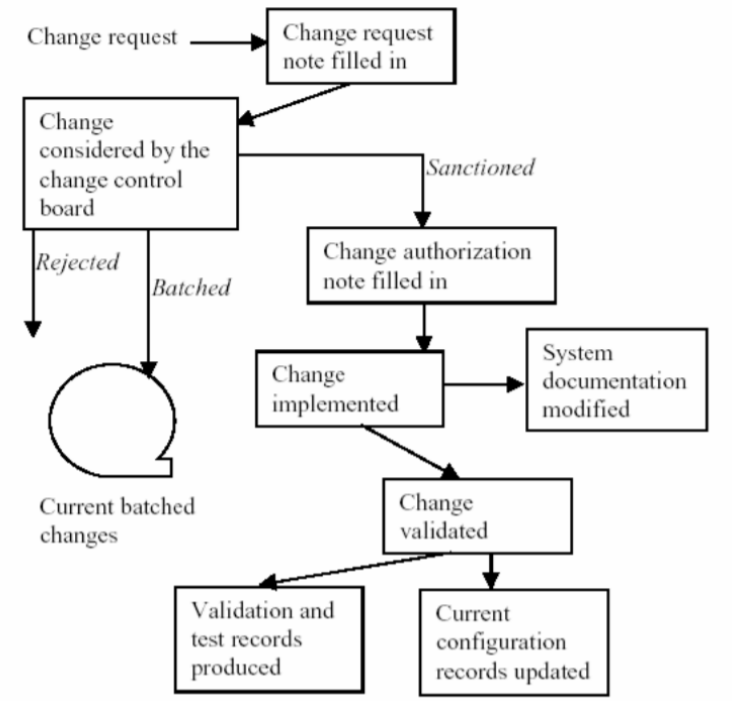

Figure 3: Ince's change process model

2.3. PRODUCT INTEGRATION STRATEGY

The project is integrated with Continuous Integration (CI) methods.

- **Develop** describes the practices necessary to implement stories and commit the code and components to version control.
- **Build** describes the practices needed to create deployable binaries and merge development branches into the trunk.
- **Test end-to-end** describes the practices necessary to validate the solution.
- **Stage** describes the practices necessary to host and validate the solution in a staging environment before production.

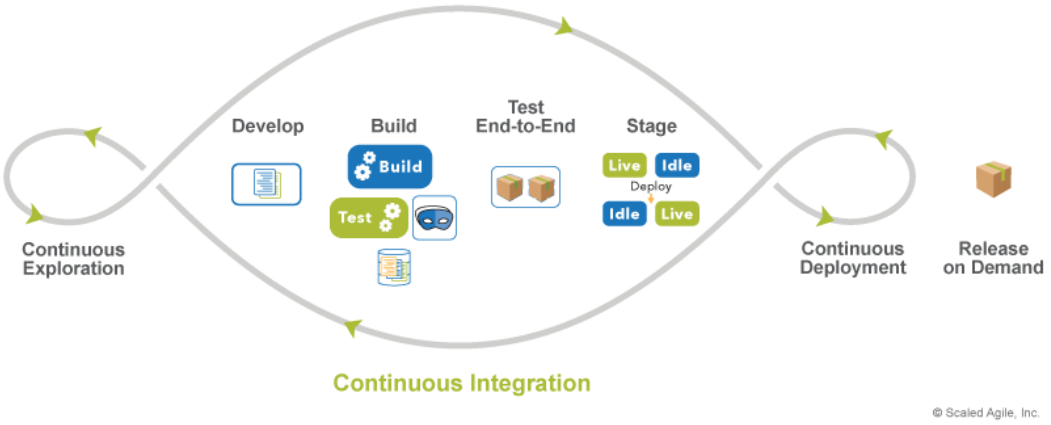

Figure 4: Continuous Integration

We decided to implement the CI method for EDSDA with Bitbucket Pipeline. Bitbucket Pipelines is an integrated CI/CD service, built into Bitbucket. It allows users to automatically build, test, and even deploy code based on a configuration file in their repository.

2.4. QUALITY MANAGEMENT

2.4.1. Estimates of Defects to be detected

Pre-release review defects

Process	Planned found by review	Actual found by review
Requirement	- Change business requirement - Change project architectural design requirements - Shortcomings in defining use cases and user stories	- Change business requirement - Change project architectural design requirements - Shortcomings in defining use cases and user stories - Change requirements on data processing and storage methods
Amount of work	- A lot of work causes overload for programmers who have a job	- A lot of work causes overload for programmers who have a job
Programming tools	- Inappropriate use of programming tools	
Technologies stack	- Inappropriate use of technologies stack - Some technologies are no longer supported - Lack of new technology	- Inappropriate use of technologies stack - Some technologies are no longer supported - Lack of new technology - Difficulty in accessing paid technologies
Reference studies	- Shortage of research articles related to the system - Articles often have a large academic volume and are difficult to apply in practice	- Shortage of research articles related to the system - Articles often have a large academic volume and are difficult to apply in practice - Difficult to access internal research works

Process	Planned found by review	Actual found by review
Design	- Old version UI is hard to reuse to extend the functionality	- Old version UI is hard to reuse to extend the functionality - Completely change the UI of the system
Automatic processing method	- Lack of supporting technologies and tools	- Lack of supporting technologies and tools - Data processing results are not quite as expected
Coding	- Difficult to implement - Use different languages for each module	- Difficult to implement - Use different languages for each module - Dirty code
System integration	- The difficulty of creating a fully automated pipeline to run different modules - It is very difficult to integrate quickly because the system is built on many different programming languages and technologies	- The difficulty of creating a fully automated pipeline to run different modules - It is very difficult to integrate quickly because the system is built on many different programming languages and technologies
Deliver on time	- Large workload leads to late deadline - Difficulty in implementation causing slow progress	- Large workload leads to late deadline - Difficulty in implementation causing slow progress
Other	- Lack of money to maintain the cloud server and organize weekly meetings	- Lack of money to maintain the cloud server and organize weekly meetings
Total	**18**	**24**

2.4.2. Strategy for Meeting Quality Objectives

Strategy	Expected Benefits
Use standards and design patterns to implement, prevent unnecessary defects during system installation	5-10% reduction in defect injection rate and about 2% improvement in productivity
Public project to perform public user acceptance test, ensure test samples are verified with a large number of users to evaluate system performance and find potential defects	Find out most defects in UI/UX and data processing, take corrective measures as soon as possible, through which can improve the system quality in the best way
Write as many test cases as possible, make sure to cover all test cases	Ensure that the testing process is continuous without being forced at the end of the project, increasing the quality of each module of the system.
Use rolling wave planning to execute test cases continuously during development	Approximately 5% reduction in defect injection rate and 1% improvement in overall productivity
Use proven technologies, absolutely do not use tools that are no longer supported	Ensure the continuity and scalability of the system, avoiding the case of sudden system death when technologies stop supporting

2.4.3. Quality Control

Review Item	Type of Review	Reviewer	When
System Requirement Specification	One-person review	Scrum Master	Before the project kick-off meeting
Scope, Objective, and Goal	One-person review	Scrum Master	End of requirements 90%
System architecture specification	Group review	Scrum Master, Developer	After finishing System Requirement Specification
UI/UX Design	Group review	Scrum Master, Developer	After finishing the System architecture specification

Review Item	Type of Review	Reviewer	When
Project Plan, Scrum Schedule	Group review	Scrum Master, Developer, Tester	After finishing the System architecture specification
Resource allocation	One-person review	Scrum Master	After finishing Project Plan
Design document, object model	Group review	Scrum Master, Developer, Tester	End of 90% design
Database design	Group review	Scrum Master, Developer	After finishing object model design
Testing plan	Group review	Scrum Master, Tester	End of Sprint 0
Code	Group review	Scrum Master, Developer	After each module built

2.4.4. Measurements Program

Data to be collected	Purpose	Responsible	When
Size: No. of KLOC// FP	Group review or One-person review	PM/SM	At the end of stages
Effort: No. person-day		Team members	Daily
Quality: No. defects detected		Reviewer, Tester	Right after the review/test
Schedule		PM/SM	Weekly and at the end of stages

2.5. UNIT TESTING STRATEGY

In this project, most of the functionality is highly specific, so we focus on using manual testing. Besides, there are still some functions that are repeated over and over again, for example:

- Import data source.
- Auto ETL process.
- Auto-generate RDF Data Cubes
- Export report.

We use automation tests with functions that are repeated above.

Tool:

- Manual Test**:** Google Sheet, Trello**.**
- Automation Test: Cypress.

There are principles while doing unit tests:

- For each class, there should be a test class that tests all the public methods.
- Tests cover at least positive tests and negative tests.
- Dependencies to other classes should be substituted by mock objects.
- Each test case covers exactly one functionality to achieve a quick bug fixing.
- For each abstract class, an abstract test class will be implemented. This abstract test class tests the implementation parts of the abstract class and outlines the correct use of the abstract class and the test classes to implement for the concrete classes.

Requirements:

- At least 70% of lines of code get coverage.
- Unit tests must be done once before the tester executes test cases in each sprint.
- Unit tests must be performed consistently, ensuring that there is no fluctuation in the number of unit tests per module.

2.6. INTEGRATION TESTING STRATEGY

Testers must design a test plan for integration testing with both Sub-component integration testing and Component integration testing because a component is typically composed of a set of sub-components and these sub-components consist of classes and packages.

- Sub-component integration testing: The integration tests between sub-components ensure compatibility of these software units across development cycles. An example of a sub-component integration test in the

Design Environment is the validation of outputs from the Modelling Library against the expected inputs of the Modelling Tool.

- Component integration testing: These tests will examine the compatibility of components' APIs according to a predefined set of rules. Furthermore, they examine the basic functionalities of a group of components as a sub-system. An example of such tests is between Design and Algorithm Optimization environments, where the outputs of the Modelling Tool will be tested against expected inputs of the Optimization Tool.

2.7. SYSTEM TESTING STRATEGY

For system testing, we use the following technique:

- Usability Testing: Test focus on user-friendly, easy-to-interact interface, flexibility in handling controls, and ability of the system to meet project objectives.
- Load Testing: Check whether the performance of the system is balanced and stable, whether the criterion is met in quality attribute or not.
- Functional Testing: provide missing functionality, functions list that testers think are likely to affect the product.

3. ESTIMATE

3.1. SIZE

3.1.1. Lines of code:

Modules	LOC
UI	1300
Database Design	550
Data Clean Engine	900
ETL Process	1000
RDF Automate Generator	800
GraphDB API	600
CI/CD Pipelines	200
Total	**5350**

3.2. EFFORT

Planned Resources				
	Min man-days	**Max man-days**	**Quantity**	**Position Title**
Total Development	133	200	3	Software Engineer
Total QA	30.5	44	1	Tester
Total Data Science	40	45.5	2	Data Engineer
Total UI/UX Design	20	25	1	Designer
Total UI Implementation	40	45	1	Front-end Developer
Total BA	12	16	1	Business Analyst
Total PM	30	44	1	Project Manager
Total CI/CD	20	30	1	DevOps

3.3. SCHEDULE

3.3.1. Project Milestone & Deliverables

Project Milestone

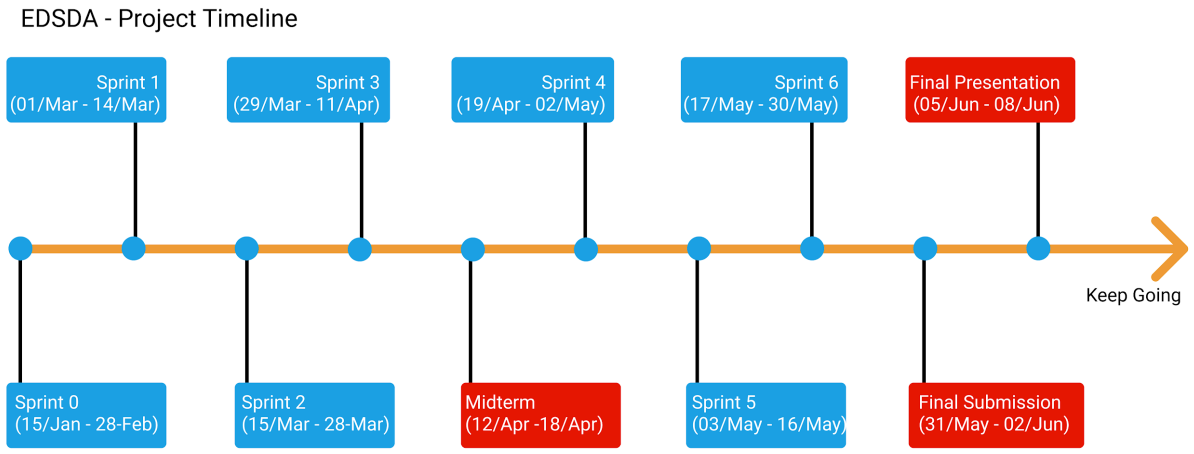

Figure 3.1: Project Timeline

DELIVERABLES

No	Activities	Deliverables
0	Requirements Description	Requirements Description 2.1
1	Project Proposal	Project Proposal Document 2.1
2	Product Requirement	Product Requirement Document 2.1
3	Project Plan	Project Plan Document 2.1
4	Product Backlog	Product Backlog Document 2.1
5	Architecture Document	Architecture Document Document 2.1
6	Database Design	Database Design Document 2.1
7	Interface Design	Interface Design Document 2.1
8	Test Plan	Test Plan Document 2.1
9	Test Case	Test Case Document 2.1
10	Acceptance Criteria	Acceptance Criteria Document 2.1
11	Sprint Backlog & Burndown Chart	Sprint Backlog & Burndown Chart Document 2.1

No	Activities	Deliverables
12	Team Reflection	Team Reflection Document 2.0
13	Technologies Stack	Technologies Stack Document 2.0

3.3.2. Work Breakdown Structure

WBS NUMBER	TASK TITLE
1	Preparation
2	Data Modeling
3	Physical Warehouse Design
4	Initiate Dynamical Data Warehouse
5	Finding Data Sources
6	Data processing
7	ETL Process Validating
8	Automate ETL Setting
9	Expanding database
10	Crawl data
11	Building Data Cubes Importing method
12	Building additional features
13	Common RDF Data Cubes Structure Designing
14	Setting up Automate DW to RDF Process Tool & Environment
15	Implementing Automate DW2RDF Process
16	Setting up RDF Data Cubes Storing & SPARQL Endpoint
17	Validating RDF Data Cubes
18	Building SPARLQL_Rest API
19	Building UI
20	Testing
21	Integrate
22	Deploy
23	Release

3.3.3. Detailed Schedule

WBS	Task	Start	End	Days
	Sprint 1	**01-Mar-21**	**14-Mar-21**	**14**
1	Crawl data			
2	Clean data			
3	Initiate RML process			
4	Initiate MIRROR			
5	Initiate GraphDB Database			
6	UI Design			
7	Initiate & config React project			
8	Testing			
	Sprint 2	**15-Mar-21**	**28-Mar-21**	**14**
1	Create DW			
2	ETL Process using a tool			
3	Define a common mapping for RML & MIRROR			
4	Implement API for importing data, getting data from GraphDB			
5	Building UI			
6	Testing			
	Sprint 3	**29-Mar-21**	**11-Apr-21**	**14**
1	Getting, and Cleaning imported data			
2	ETL the imported data			
3	Describe common schema for RDF Data Cubes			
4	Run RML & MIRROR with sample data			
5	Integrate sample UI & API			
6	Building UI			
7	Testing			
	Midterm	**12-Apr-21**	**18-Apr-21**	**7**
1	Deploy version 0.1			
2	Review documentation			

WBS	Task	Start	End	Days
	Sprint 4	**19-Apr-21**	**02-May-21**	**14**
1	Automating clean data			
2	Schedule for ETL data pipeline			
3	Customize RML & MIRROR			
4	Implement pipeline for automating RDF Data Cube generation			
5	Building UI			
6	Testing			
	Sprint 5	**03-May-21**	**16-May-21**	**14**
1	Handling new importing data to DW(Dimensions, Measures)			
2	Automating ETL for new importing data			
3	Integrate automate ETL & RDF generation			
4	Automating detect dimension & measure			
5	Building UI			
6	Testing			
	Sprint 6	**17-May-21**	**30-May-21**	**14**
1	Automating crawling data from NGO, or any new defined data source.			
2	Cleaning data process for the new defined data source.			
3	Set up VPS for deploying			
4	Migrate GraphDB to VPS			
5	Building UI			
6	Integration			
7	Testing			
	Final	**31-May-21**	**08-Jun-21**	**9**
1	Review documentation			
2	Release / Deploy final version			
3	Presentation			

3.4. RESOURCE

3.4.1. Human Resources:

Position Titles	Quantity	Member
Backend Developer	2	Dong Ky, Hoa Vo
Frontend Developer	1	Tin Pham
Data Engineer	2	Dong Ky, Hoa Vo
Graphics Designer	1	Tin Pham
Business Analyst	1	Hoa Vo, Tin Pham
Tester	1	Kieu Tran
Project Manager	1	Hoa Vo
DevOps	1	Hoa Vo
Total Human-Resources	**10**	

3.4.2. Equipment

Name of equipment	Quantity
Laptop	4
Monitor	4
VPS Server	1
Tablet	1

3.5. INFRASTRUCTURE

Work/Product	Purpose	Expected Availability by	Note
Development Environment			
Win 10, Linux	Operating Systems	Initiation stage	
Postgresql	DBMS	Initiation stage	
Javascript	Development language for Web interface and API server	Initiation stage	
Python	Development language for data processing, Data warehouse generating, crawl data, and handling data sources.	Initiation stage	
Hardware & Software			
Cloud server	2vCPUs, 1GB memory, 10GB storage, ubuntu v20.04 minimal		
Figma	UI Design & Architecture design		
Other Tools			
Github, bitbucket	Source version control	Initiation stage	
Cypress	Unit Test	Construction stage	
Googlesheet	Effort logging	Initiation stage	
Slack	Discussion	Initiation stage	
Trello	Task tracking	Initiation stage	
Google Office	Managing documents	Initiation stage	
Discord	Meeting online	Initiation stage	

3.6. TRAINING PLAN

Training Area	Participants	When, Duration	Waiver Criteria
Technical			
Python Language		7 days	If already trained
Javascript	Kieu	7 days	if already trained
Business domain			
Environment		7 days	
Data Visualization	Tin, Hoa, Dong	7 days	
Process			
Quality system	All members	3 hrs	If already trained
Configuration management	All members	2 hrs	If already trained for CC. For others, on-the- job training
Group review	All members	4 hrs	If already trained
Defect prevention	All members	4.5 hrs	Mandatory

3.6.1. Compulsory training program:

- RDF Data Cubes Introduction courses.
- Data Warehouse in Advanced.
- Automate ETL tutorial.
- ReactJS DnD.
- ReactJS Flow.
- Redux.
- Testing Techniques Course.
- Automation Testing tutorial.
- How to apply CI/CD with Bitbucket Pipeline.

3.6.2. Additional training program:

- The way of working on a Scrum project.
- Release Planning Meeting
- Sprint Planning Meeting
- Daily Scrum Meeting
- Sprint Review Meeting
- Retrospective
- Scrum of Scrums

3.7. FINANCE

COST DESCRIPTION	DETAIL
Salary	Duration: 16 weeks (112 days) Man-hour: 5hours/1day Salary: $2/1 hour Persons: 4 members Overtime cost per hour: 1.5$/hour The salary of 1 person: $1000 **Total: $4000**
Laptop	Laptop for each member: $800 **Total: $3200**
Monitor	LG Monitor for each member: $100 **Total: $400**
VPS Engine server	VPS Engine server per month on Google Cloud: $100 VPS Engine server for 3 months: 60$ **Total: $300**
Maintenance	Cost per month: $400
Bugs fixing	Cost per bug: $20
Total	Salary + Laptop + Monitor + VPS Engineer server **Total: $7900**

4. PROJECT ORGANIZATION

4.1. ORGANIZATION STRUCTURE

Role	Responsibility	Name
Product Owner	Understand the user and customers with their needs. Collaborate with the development team. Manage the stakeholders. Describe the user experience and product features. Provides detailed user stories.	PhD Binh Thanh Nguyen
Scrum Master	Communicate the value of Scrum Teach the organization on Scrum to maximize business value Facilitate Sprint Planning, Daily Scrums, Sprint Reviews, and Retrospective Meetings Create the Task Board and Sprint Burndown Chart at the start of every Sprint Attend all Scrum meetings Preserve the integrity and spirit of the Scrum framework Maintain the focus of the Team Make the Team aware of impediments and facilitate efforts to resolve them Serve as a coach and mentor to members of the Team Respectfully hold the Team, Product Owner, and Stakeholders accountable for their commitments Continually work with the Team and business to find and implement improvements	Hoa, Vo Van
Secretary	Record the content of group meetings and activities of the member	Tin, Pham Van
Reviewer	Analysis of the functions and requirements of the product. Review documents related to the project	Hoa, Vo Van
Team Leader	DevOps, Back-end Dev: Server, RDF Data Cubes, B.A	Hoa, Vo Van
Team member	Frontend Developer, UI/UX Designer, DevOps	Tin, Pham Van
Team member	Back-end Dev: Database, Crawl	Dong, Ky Huu
Team member	Frontend Developer, Tester	Kieu, Tran Thi Thanh

4.2. PROJECT TEAM

Full Name	Email	Phone number	Role
Hoa, Vo	hoavo.dng@gmail.com	0935.193.182	Scrum master
Tin, Pham Van	tinphamvan123@gmail.com	0932.535.175	Team member
Dong, Ky Huu	kyhuudong@gmail.com	0898.246.980	Team member
Kieu, Tran Thi Thanh	thanhkieutran391@gmail.com	0358.583.251	Team member

4.3. EXTERNAL INTERFACES

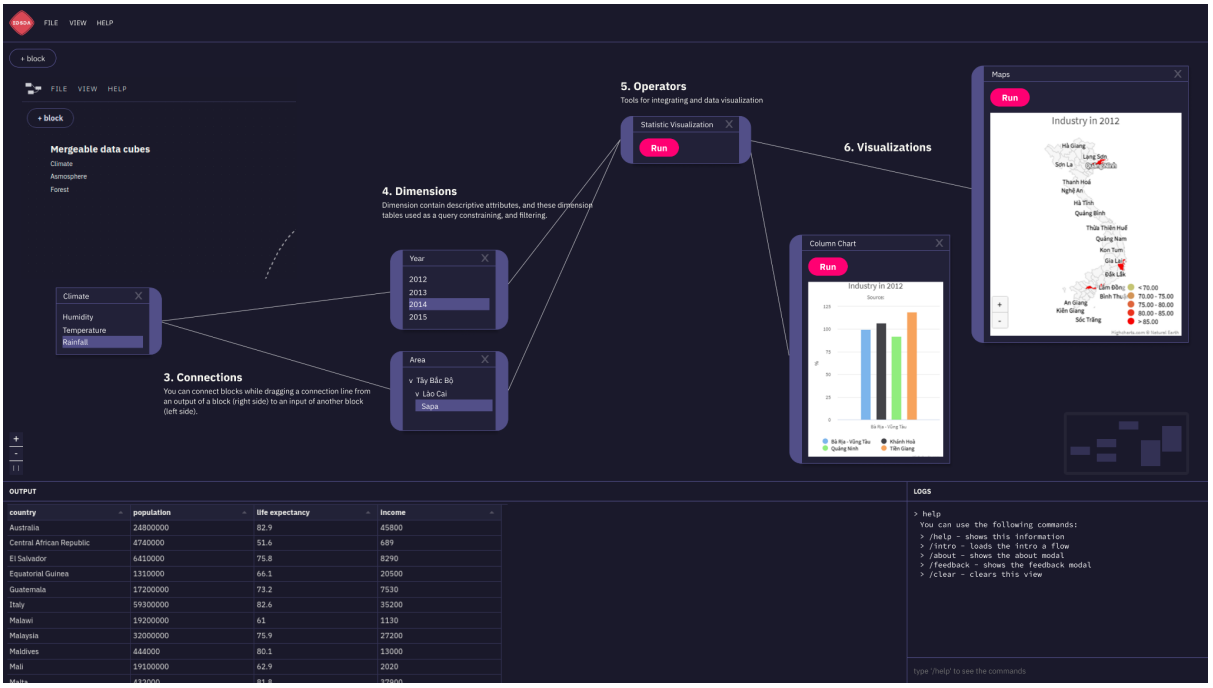

Figure 5: UI Design of EDSDA

5. COMMUNICATION & REPORTING

5.1. REPORTING METHODOLOGY

Audience/ Attendees	Topic/ Deliverable	Frequency	Method
- Product Owner - Scrum Master - Team Members	Project Progress Review	Weekly	Email, Skype
- Product Owner - Scrum Master - Team Members	Explicit Requirement	When needed	Email, Skype
- Mentor - Scrum master - Team members	Milestone review	End of each Milestone	Skype
- Scrum master - Team members	Daily tasks	Each day	Slack, Discord, Trello

5.2. COMMUNICATION METHODOLOGY

Type of Communication	Communication Schedule	Communication way	Who Initiates	Recipient
Status Report (Daily meeting)	Daily at 10AM	Slack	Scrum Master	Scrum Team
Schedule and Effort Tracking	Daily	Google Sheets, Trello	Scrum Master	Scrum Team
Work Review	Daily at 10AM	Trello	Scrum Master	Scrum Team
Work Report	Every Sunday	Slack or Face to Face	Scrum Master	Scrum Team
Project Review, ask problems	Every Monday (flexible)	Skype	Scrum Master	Scrum Team, Mentor
Ask & Review problems	Anytime	Face to face, Slack, Discord, Skype	Scrum's Member	Mentor and Scrum Team

6. CONFIGURATION MANAGEMENT

No	Tool	Content
1	Google Sheet	Track member activities. At the end of each day, team members will post on time log and Scrum Master will check.
2	Google Document	Track the changing of documents & manage versions of documents.
3	Bitbucket	Repositories for source code version management.
4	Weekly Meeting	Hold a meeting every week to assign tasks to each member. If there are some emergencies but we cannot sit together then we can use Discord to discuss online.
5	Document	All meetings must be documented and pictured.
7	Slack	Store document resources and designed components, Daily Scrum
8	Discord	Discuss online, stream and share problems
9	Github	Repositories for open-source code of the project

7. SECURITY ASPECTS

To deal with security risks, EDSDA is precisely process-configured at every stage of the project.

- Concept and planning

The purpose of this stage is to define the application concept and evaluate its viability. This includes developing a project plan, writing project requirements, and allocating human resources.

    - Prepare a list of security requirements for the project.
    - Training sessions provide essential security knowledge ranging from basic threat awareness to in-depth information on secure development.
- Architecture and design

The purpose of this stage is to design a product that meets the requirements. This includes modeling the application structure and its usage scenarios, as well as choosing third-party components that can speed up development. The result of this stage is a design document.

    - Secure design :The design document and subsequent updates are validated in light of the security requirements. Early design reviews assist in identifying features exposed to security risks before they are implemented.
- Implementation
    - This is the stage at which an application is actually created. This includes writing the application code, debugging it, and producing stable builds suitable for testing.
    - Guides and checklists remind programmers of typical mistakes to be avoided, such as storing unencrypted passwords. Enforcing secure coding principles eliminates many trivial vulnerabilities and frees up time for other important tasks.
    - Manual code reviews are still a must for building secure applications. Timely reviews help developers to flag and fix potential issues before they shift attention to other tasks.
- Testing and bug fixing

The purpose of this stage is to discover and correct application errors. This includes running automatic and manual tests, identifying issues, and fixing them.

We use the rolling wave testing method to perform continuous testing after each phase, which ensures that security issues will be detected and resolved as soon as possible.

- Release and maintenance

At this stage EDSDA goes live, with many instances running in a variety of environments.

Before going live, we set up security for the cloud server, install SSL for hosting and related procedures to manage the cloud environment as closely as possible.

8. REFERENCE

[EDSDA]Proposal: https://docs.google.com/document/d/1i5YAMZ1wxw7DIX-KArDY1_1yXWSKtZ7MliIYvf_uAoo/

What is Scrum: https://www.scrum.org/resources/what-is-scrum

Design: https://www.figma.com/

Security Aspect: https://www.ptsecurity.com/ww-en/analytics/knowledge-base/how-to-approach-secure-software-development/

Integrating Quality Management System into Software Development Processes:

https://assist-software.net/blog/integrating-quality-management-system-software-development-processes