Process MeNtOR 3.0

Uni-SEP

McBal Demographic Analysis Application (MDAA) **Design Document**

Version:	1.2.1
Print Date:	
Release Date:	
Release State:	Initial
Approval State:	Draft
Approved by:	
Prepared by:	Henry So, Jacob Chun, Samuel Su, Yan Qing Niu
Reviewed by:	
Path Name:	
File Name:	ImplementationDoc.pdf
Document No:	1

Document Change Control

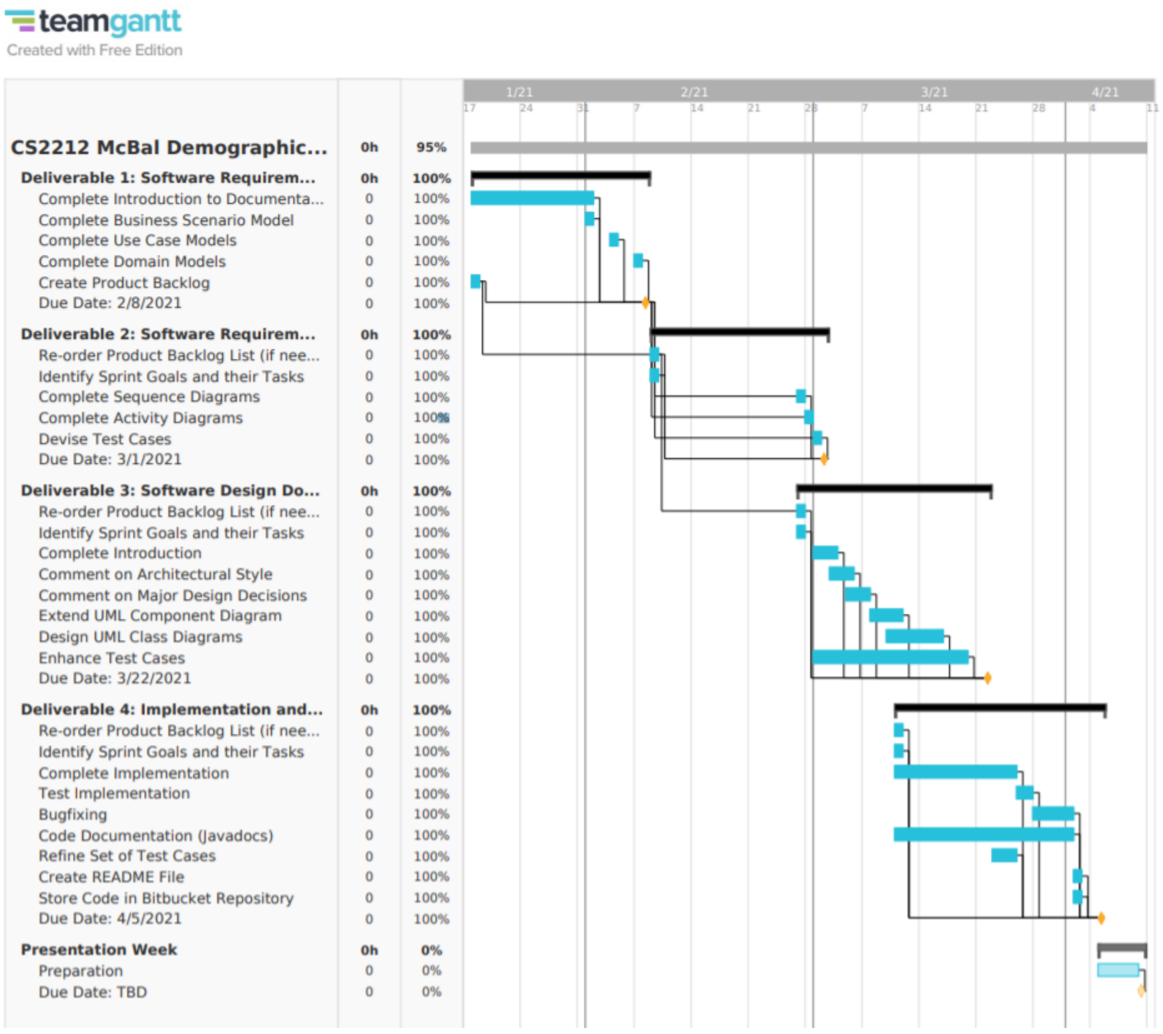
Version	**Date**	**Authors**	**Summary of Changes**
1.0	3/28/2021	Henry So, Jacob Chun, Samuel Su, Yan Qing Niu	Edits to product backlog and sprint backlog and activity plan
1.1	3/31/2021	Henry So, Jacob Chun, Samuel Su, Yan Qing Niu	Revised test cases to encompass programs functionality
1.2	4/4/2021	Henry So, Jacob Chun, Samuel Su, Yan Qing Niu	Completed the Test Driven Development section of the document. Final edits were made.

Document Sign-Off

Name (Position)	**Signature**	**Date**
Henry So	HS	4/5/2021
Jacob Chun	JC	4/5/2021
Samuel Su	SS	4/5/2021
Yan Qing Niu	DN	4/5/2021

Contents

1.1 Gantt Chart



CS2212 McBal Demographic...	**0h**	**95%**
Deliverable 1: Software Requirem...	**0h**	**100%**
Complete Introduction to Documenta...	0	100%
Complete Business Scenario Model	0	100%
Complete Use Case Models	0	100%
Complete Domain Models	0	100%
Create Product Backlog	0	100%
Due Date: 2/8/2021	0	100%
Deliverable 2: Software Requirem...	**0h**	**100%**
Re-order Product Backlog List (if nee...	0	100%
Identify Sprint Goals and their Tasks	0	100%
Complete Sequence Diagrams	0	100%
Complete Activity Diagrams	0	100%
Devise Test Cases	0	100%
Due Date: 3/1/2021	0	100%
Deliverable 3: Software Design Do...	**0h**	**100%**
Re-order Product Backlog List (if nee...	0	100%
Identify Sprint Goals and their Tasks	0	100%
Complete Introduction	0	100%
Comment on Architectural Style	0	100%
Comment on Major Design Decisions	0	100%
Extend UML Component Diagram	0	100%
Design UML Class Diagrams	0	100%
Enhance Test Cases	0	100%
Due Date: 3/22/2021	0	100%
Deliverable 4: Implementation and...	**0h**	**100%**
Re-order Product Backlog List (if nee...	0	100%
Identify Sprint Goals and their Tasks	0	100%
Complete Implementation	0	100%
Test Implementation	0	100%
Bugfixing	0	100%
Code Documentation (Javadocs)	0	100%
Refine Set of Test Cases	0	100%
Create README File	0	100%
Store Code in Bitbucket Repository	0	100%
Due Date: 4/5/2021	0	100%
Presentation Week	**0h**	**0%**
Preparation	0	0%
Due Date: TBD	0	0%

1.2 Project Backlog and Sprint Backlog

1. Create login system
    1.1. Create the LoginService interface
    1.2. Create the LoginService
    1.3. Create the VerificationServer interface
    1.4. Create the VerificationServer
    1.5. Create the CheckDatabase interface
    1.6. Create the LoginDatabase class (using plain text file)
    1.7. Create initial registration system
2. Create main application UI
    2.1. Create the Launch main UI interface
    2.2. Create the UI refresh methods
    2.3. Create the onPress methods

    2.4. Create list of countries selectable
    2.5. Create year drop-down menus
    2.6. Create viewer drop-down menu
    2.7. Implement buttons
3. Create the Populator Component
    3.1. Create the Populator class
    3.2. Implement the SelectionObject class
    3.3. Create interface to populate SelectionObjects
4. Create the Analysis Component
    4.1. Create the compute server class exposing a recalculate interface
    4.2. Create the Factory Method
        4.2.1. Create a Creator class
        4.2.2. Create the 8 analysis creators
        4.2.3. Create a analysis object class exposing a calculate interface
        4.2.4. Create the 8 analysis types
    4.3. Create the Data class exposing a FindData interface
    4.4. Create the ResultObject class exposing CreateResult interface
5. Create data retrieval API component
    5.1. Create CreateData class with ability to make API calls, implementing the Search interface
6. Create the Viewer Components
    6.1. Create the Observer class that implements SendUpdate interface
    6.2. Create the DisplayViewer logic class that handles program execution
    6.3. Create the ViewerCreation interface
    6.4. Create the 5 ViewerType child classes that implement the interface
7. Test application with test cases
8. Debug potential errors/bugs
9. Upload code to BitBucket
10. Create README file

1.3 Group Meeting Logs

Present Group Members	**Meeting Date**	**Issues Discussed / Resolved**
Henry So, Jacob Chun, Yan Qing Niu, Samuel Su	March 26/2021 10:00 PM - 2:00 AM	- Revised sprint backlog, product backlog - Discussed the overall plan for the implementation - Started implementing front end of the application and login system - Uploaded pull requests to master branch on BitBucket for respective code changes - Debugged any errors with main UI

Henry So, Jacob Chun, Yan Qing Niu, Samuel Su	March 30/2021 6:00 PM - 12:00 AM	- Implementing analysis, selection classes and reader classes - Uploaded pull requests to master branch on BitBucket for respective code changes - Debugged errors involved with analysis classes and reader classes
Henry So, Jacob Chun, Yan Qing Niu, Samuel Su	April 1//2021 2:00 PM - 8:00 PM	- Implemented viewer classes - Uploaded pull requests to master branch on BitBucket for respective code changes - Debugged errors involved with viewer classes
Henry So, Jacob Chun, Yan Qing Niu, Samuel Su	April 4//2021 6:00 PM - 12:00 AM	- Implemented - Included javadocs documentation (comments) - Updated gantt chart - Completion of the Test Driven Development section - Final edits made before submitting the document. - Completed README file

2 Test Driven Development

Initial test cases will be provided in the form of a table as follows:

Test ID	1.1
Category	Login System / Verification
Requirements Coverage	UC1-Successful-User-Login
Input Data	Username → User Password → Pass
Procedure	1. The User opens the system 2. The User provides a user name to the login system 3. The User provides a password to the login system 4. The verification server checks to see if the user's credentials are valid

	5. The verification server invokes the loginCheck method from the LoginDatabase to see if password matches with the given username 6. The User credentials are correct 7. The User is presented with the main UI window
Expected Outcome	Login form closes, and the user is presented with the main UI window
Notes	The User logins successfully and the methods invoked to check the User credentials are login(username,password) from the LoginService class, checkPin(username, password) from the VerificationServer class, and loginCheck(username) from the LoginDatabase class. Once the User log in successfully, the valid instance variable is set to True. Classes involved: LoginService, VerificationServer, LoginDatabase

Test ID	1.2
Category	Login System / Verification
Requirements Coverage	UC1-Unsuccessful-User-Login
Input Data	Username → Failed Password → Test
Procedure	1. The User opens the application 2. The User provides a user name to the login system 3. The User provides a password to the login system 4. The verification server checks to see if the user's credentials are valid 5. The verification server invokes the loginCheck method from the LoginDatabase to see if password matches with the given username 6. The User credentials are incorrect 7. Incorrect credentials message appears on the screen 8. The application terminates through usage of a method from LoginSystem, applicationExit()
Expected Outcome	Credentials cannot be verified and application terminates.
Notes	The User logins unsuccessfully and the methods invoked to check the User credentials are login(username,password) from the LoginService class, checkPin(username, password) from the VerificationServer class, and loginCheck(username) from the LoginDatabase class. Classes involved: LoginService, VerificationServer, LoginDatabase

Test ID	2.1

Category	Analysis Selection
Requirements Coverage	UC2-Successful-Analysis-Select
Input Data	User clicks appropriate analysis type (eg. Average Expenditure on Education (% of GDP))
Procedure	1. User logged in 2. User selects analysis type
Expected Outcome	The system confirms the user's selection and stores the chosen analysis type.
Notes	Country selection and year selection is dependent on analysis selection. The User populates the SelectionObject. When the analysis type is chosen, the SelectionObject invokes the setAnalysisType(analysisType) method and passes in the analysis type chosen by the User. Classes involved: MainUI, Populator, SelectionObject

Test ID	3.1
Category	Country Selection
Requirements Coverage	UC3-Successful-Country-Select
Input Data	Analysis type → Average Expenditure on Education (% of GDP) Country → USA
Procedure	1. User logged in 2. User selected analysis type 3. User selects allowed country 4. System stores country selection
Expected Outcome	The system confirms the user's selection and stores their country. The drop down menu reflects the user's selection.
Notes	Available countries are determined by the engineers. Upon successful country selection, the User populates the SelectionObject. When the country is chosen, the SelectionObject invokes the setCountry(country) method and passes in the country chosen by the User. Classes involved: MainUI, Populator, SelectionObject

Test ID	3.2
Category	Country Selection
Requirements Coverage	UC3-Unsuccessful-Country-Select
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → Ethiopia

Procedure	1. User logged in 2. User selects analysis type 3. User selects unavailable country 4. System prints error message for user
Expected Outcome	System does not allow country to be selected and prints an error message for the user
Notes	Unavailable countries are determined by the engineers. The method errorMsg(errorStr) will be invoked and a window will pop up, alerting the user that the country was unsuccessfully selected. Classes involved: MainUI

Test ID	4.1
Category	Year Selection
Requirements Coverage	UC4-Successful-Year-Select
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → United Kingdom Start Year → 2016 End Year → 2018
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start year 5. User selects end year 6. System stores start and end year selection and checks the time period in between to see if it is valid for the analysis type chosen
Expected Outcome	The system confirms the user's selection and stores their start and end year. The drop down menu reflects the user's selection.
Notes	The user successfully populates the variable start and end of the SelectionObject when selecting a suitable time period in which analysis is available. Invokes the setStart(start) and setEnd(end) method and passes in the start and end years that the user selects. Classes involved: MainUI, Populator, SelectionObject

Test ID	4.2
Category	Year Selection
Requirements Coverage	UC4-Unsuccessful-Year-Select
Input Data	Analysis Type → CO2 Emissions vs Energy Use vs PM2.5 Air Pollution Country → USA Start Year → 1978

	End Year → 1980
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start year 5. User selects end year 6. System stores start and end year selection and checks the time period in between to see if it is valid for the analysis type chosen 7. The time period is invalid for the analysis type chosen 8. Error message is displayed 9. User is prompted to reselect a new start year and end year
Expected Outcome	The user is prompted by the system to reselect a new start year and end year that is valid for the analysis type chosen.
Notes	The method errorMsg(errorStr) will be invoked and a window will pop up, alerting the user that the years were unsuccessfully selected. Classes involved: MainUI, Populator

Test ID	5.1
Category	Additional / Removal of Viewers
Requirements Coverage	UC5/6-Successful-Viewers
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → United Kingdom Start Year → 2016 End Year → 2018 Viewer → Pie Chart
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start and end year 5. User selects viewer from drop-down menu 6. User selects to add viewer a. Addition of viewer successful b. Removal of viewer successful
Expected Outcome	If the user adds the viewer successfully, the system should store the type of viewer to be displayed. If the user removes the viewer successfully, the system should remove the type of the viewer from storage.
Notes	The user successfully populates the viewers array. Each successive viewer added is put into the array. Upon the population of the viewersArray, variable viewers is set with the method call setViewers(viewers). Classes involved: MainUI, Populator, SelectionObject

Test ID	5.2
Category	Additional / Removal of Viewers
Requirements Coverage	UC5/6-Unsuccessful-Viewers
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → United Kingdom Start Year → 2016 End Year → 2018 Viewer → Pie Chart User presses "+" User presses "+"
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start and end year 5. User selects viewer from drop-down menu 6. User selects to add or remove viewer a. Addition of viewer unsuccessful b. Removal of viewer unsuccessful 7. Error message is displayed
Expected Outcome	If the user adds the viewer unsuccessfully, then the system already has the viewer stored. If the user removes the viewer unsuccessfully, the system does not have the viewer stored yet. In both cases, an error message is displayed -- The method errorMsg(errorStr) will be invoked and a window will pop up, alerting the user that the viewers were unsuccessfully selected.
Notes	Classes involved: MainUI, Populator, SelectionObject

Test ID	7.1
Category	Data Procurement
Requirements Coverage	UC7-Successful-Data-Retrieval
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → United Kingdom Start Year → 2016 End Year → 2018 User presses "Recalculate"
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start and end year 5. User presses "Recalculate" button 6. System takes selection and requests from Web Service 7. Web Service sends necessary data to System

	8. Data is usable 9. Data is processed to be used 10. Data is stored
Expected Outcome	Based on the selection the user made, the system requests and receives data from the web service to be processed and stored.
Notes	Once the user has populated the SelectionObject fully, the ComputeServer class handles it and implements a factory design based on the type of selection. It will create the required AnalysisCreator, and then build the Analysis(1...8) object based on the analysis type selected. Subsequently, the AnalysisObject goes to the Data class and implements the facade design pattern, where the data from this class will eventually create an object from the CreateData class, where the data requests are made. The CreateData class invokes the World Bank Repository API through the usage of method invokeAPI(sendString) and the data from the JSON file returned is then processed and a ResultObject is made from the contents of the JSON file. Classes involved: MainUI, AnalysisObject, Data, CreateData, ResultObject, Populator

Test ID	7.2
Category	Data Procurement
Requirements Coverage	UC7-Unsuccessful-Data-Retrieval
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → USA (pre-1981) Start Year → 2016 End Year → 2017 User presses “Recalculate”
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start and end year 5. User presses “Recalculate” button 6. System takes selection and requests from Web Service 7. Web Service sends necessary data to System 8. No data received (all data is unusable) 9. Display error message, prompt user to reselect choices
Expected Outcome	An error message is displayed to prompt the user to reselect new choices as the analysis cannot be done/ data is unusable.
Notes	Classes involved: MainUI, AnalysisObject, Data, CreateData

Test ID	8.1

Category	Render Results, System UI Update
Requirements Coverage	UC8-Successful-Displaying-Graphs
Input Data	Analysis Type → Average Expenditure on Education (% of GDP) Country → United Kingdom Start Year → 2016 End Year → 2018 Viewer → Pie Chart User presses “+” User presses “Recalculate”
Procedure	1. User logged in 2. User selects analysis type 3. User selects country successfully 4. User selects start and end year 5. User selects viewer from drop-down menu 6. User selects to add/remove viewer 7. User presses the “Recalculate” button. 8. Data from Web Service is received based on selected analysis type, country, start and end years. 9. Data is processed and usable 10. User interface updates, data is rendered onto chosen viewers
Expected Outcome	The analysis is shown in graphical summaries based on viewers selected.
Notes	Taking the ResultObject, the DisplayViewer class creates the viewers based off of user selection, and renders the ResultObject into a graphical chart. The user will recognize that the graphs have been displayed successfully when DisplayViewer invokes update(). The Main UI will then update and the chosen viewers will be displayed on the screen with all the AnalysisObject that the user procured. Classes involved: MainUI, ResultObject, Observer, DisplayViewer, ViewerType(1..5)