

**Computer Science and Engineering**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Social Network Analysis with Deep Learning**

**Software Design Description (SDD)**

**Version 2.0**

Document Number: SDD-002

Team B9

Chun Zen Marc Tse (czt210)

Mehmed Mladenov (mm7277)

**REVIEW AND APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Printed Name and Title** | **Function**  **(Author,**  **Reviewer, Approval)** | **Date** | **Signature** |
| Chun Zen Marc Tse | Author | Dec 12, 2018 | Chun Zen Marc Tse |
| Mehmed Mladenov | Author | Dec 12, 2018 | Mehmed Mladenov |
| Professor Strauss | Reviewer | Dec 12, 2018 | Professor Strauss |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**REVISION LEVEL**

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision Number** | **Purpose** |
| Oct 24 2018 | Version 1.0 | Initial Release |
| Dec 12 2018 | Version 2.0 | Final Release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[1. INTRODUCTION 1](#_Toc23187)

[1.1 PURPOSE 1](#_Toc23188)

[1.2 SCOPE 1](#_Toc23189)

[1.3 IDENTIFICATION 1](#_Toc23190)

[1.4 DOCUMENT SUMMARY 2](#_Toc23191)

[1.5 SYSTEM OVERVIEW 2](#_Toc23192)

[1.6 DOCUMENT OVERVIEW 2](#_Toc23193)

[2. REFERENCE DOCUMENTS 3](#_Toc23194)

[3. SYSTEM WIDE DESIGN DECISIONS 3](#_Toc23195)

[3.1 SOFTWARE COMPONENT ARCHITECTURAL DESIGN 4](#_Toc23196)

[3.2 SOFTWARE ARCHITECTURE GENERAL DESCRIPTION 4](#_Toc23197)

[3.3 SOFTWARE ITEM COMPONENTS 5](#_Toc23198)

[3.4 COMPONENT INTERFACE IDENTIFICATION 6](#_Toc23199)

[3.5 SOFTWARE COMPONENT CONCEPT OF EXECUTION 6](#_Toc23200)

[4. SOFTWARE ITEM DETAILED DESIGN 7](#_Toc23201)

[4.1 STRUCTURE 7](#_Toc23202)

[4.1.1 Software Unit Detailed Design 7](#_Toc23203)

[4.2 STATIC RELATIONSHIP OF SOFTWARE UNIT 7](#_Toc23204)

[4.2.1 Run-time Object Instances 8](#_Toc23205)

[4.3 BEHAVIOR 8](#_Toc23206)

[4.3.1 Sequence Interaction Diagrams 9](#_Toc23207)

[4.3.2 Collaboration Diagrams 10](#_Toc23208)

[4.3.3 Activity Diagrams 11](#_Toc23209)

[4.3.4 State Diagrams 18](#_Toc23210)

[4.3.5 Event Diagrams 19](#_Toc23211)

[4.4 CONCEPT OF EXECUTION 19](#_Toc23212)

[4.5 INTERFACE DESIGN 19](#_Toc23213)

[4.5.1 Unique identifier of Interface 19](#_Toc23214)

[4.5.2 Interface Diagrams 19](#_Toc23215)

[5. IMPLEMENTATION ARCHITECTURE (NOT REQUIRED) 20](#_Toc23216)

[5.1 ALL ACTIVE AND PASSIVE CLASSES ASSIGNED TO COMPONENTS 20](#_Toc23217)

[5.2 DIAGRAMS OF PHYSICAL PACKAGING OF LOGICAL COMPONENTS 20](#_Toc23218)

[6. DEPLOYMENT ARCHITECTURE 20](#_Toc23219)

[6.1 PHYSICAL DEPLOYMENT ARCHITECTURE DIAGRAM 20](#_Toc23220)

[7. DICTIONARIES 20](#_Toc23221)

[8. SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION 21](#_Toc23222)

[9. REQUIREMENTS TRACEABILITY 21](#_Toc23223)

[9.1 SOFTWARE COMPONENT-LEVEL REQUIREMENTS TRACEABILITY 21](#_Toc23224)

[10. SYSTEM DESIGN TESTING 21](#_Toc23225)

[11. RATIONALE 22](#_Toc23226)

[12. NOTES 22](#_Toc23227)

[13. APPENDICES 22](#_Toc23228)

[13.1 DICTIONARIES 22](#_Toc23229)

[13.2 UML DIAGRAMS 26](#_Toc23230)

[IF NOT INCLUDED IN THE BODY OF THE DOCUMENT 26](#_Toc23231)

[13.3 SCHEDULE TRACKING 26](#_Toc23232)

[13.4 DEFECT TRACKING 27](#_Toc23233)

# INTRODUCTION

## Purpose

The purpose of this document is to define the contents of the Software Design Description (SDD). The SDD documents the System Architecture and the Detailed design. The document is used to communicate overall quantitative and qualitative system characteristics to operations management, technical support, training, and operators.

The intended audience for this document is mainly but not limited to, the client of the software product, the management level personnel involved with the software team, developers of the software, as well the testers and end-users of the software.

## Scope

There is a lack of simple, efficient and modularized support for determining the efficacy of marketing campaigns by operating on different data sets and performance metrics. The system will address these concerns by allowing the plugging-in of various data sets and the standards used to determine their success, as well as predict future performance. The system will provide an easy-to-use interface for this purpose, which will show results of big data analytics in visual formats, such as GIS maps, charts and spreadsheets. The system, and the algorithm that supports it, will be fine-tuned to support modularity for a variety of data sets that may be used with it as well as the performance indicators used to measure potential success. Such a system would have tremendous benefit for many organizations, simplifying the process of using Deep Learning greatly while also reducing the cost of implement such techniques.

## Identification

Social Network Analysis with Deep Learning Version 0.1.

“Software Design Documentation (SDD)” Social Network Analysis with Deep Learning, B9, Version 1.0, October 17, 2018.

## Document Summary

The document is organized in such a way to first provide a top-level description and overview of the design of software system in terms of architecture, as well as an overview of the document from section 1 to 3. In section 4-8, a more detailed description of the system is provided regarding classes used in the software, use case diagrams, event diagrams and activities diagrams. Proprietary technologies necessary to run the software successfully will also be analysed in terms of physical deployment architecture diagram and computer resources utilization. In section 9-10, the document details the manner of which requirement tracing and system testing should be conducted for the purpose of finding defects and satisfying requirements. The appendix serves as an extension to the class diagrams, as well as method of tracking schedule and defects encountered when designing the system and documenting the process.

## System Overview

The system boundary includes the software of the social network analysis itself, the boundary between end-users (researchers, data scientists, marketers in companies) and the software and the boundary between the software and external databases and data metrics from other companies (social media companies, digital marketing companies, in house data from client).

## Document Overview

Format and content

Title Page (formatted the same as standard cover)

Review/Approval Signatures

Table of Revisions (revision number, date, purpose)

Approval page (if required)

Preface (information the reader should be familiar with)

Architecture and Detailed Design

Table of Contents

List of Figures

# REFERENCE DOCUMENTS

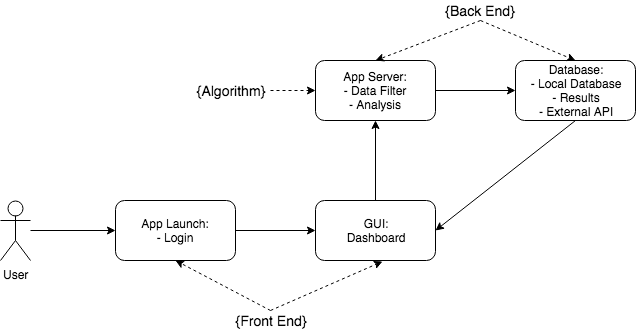
“Team B9 Project Proposal” Social Network Analysis with Deep Learning, B9, Version 1.2, September 12, 2018.

“System Requirements and Analysis Specification (RAS)” Social Network Analysis with Deep Learning, Team B9, RAS-001 Version 1.3, September 19, 2018.

“Software Project Management Plan (SPMP)” Social Network Analysis with Deep Learning, Team B9, SPMP-001 Version 1.1, October 2, 2018.

# SYSTEM WIDE DESIGN DECISIONS

## Software Component Architectural Design



## Software Architecture General Description

Three tier architecture is used for the design of system of application – Presentation, Middle and Data tier. Presentation tier includes all components of the system that directly interacts with the user. GUIs such as login and dashboard is included in the Presentation tier. Middle tier includes all component of the system that processes information passed from users and directs to the database. Algorithms such as data analysis, prediction of trend, and data filter is included in the Middle tier. Data tier includes all components of the system that deals with the management and direct control of data in own database, as well as the handling of external data from APIs of external sources. In-house database, external API, output is included in the Data tier.

## Software Item Components

*Login GUI*

The login will provide access to the dashboard, which will serve as the main primary point of access for the client in order to use the Social Network Analysis System. The login screen will require the users to enter their user credentials, and after validated, will be transported to the Dashboard.

*Dashboard*

The dashboard will provide a clear mechanism for the user to upload custom data sets in a new project to satisfy the purpose of modularity. Users can create new projects or delete existing projects. The dashboard will allow the user to continue analysis on a previous project and will clearly present the user configurable metrics for an analysis they have ran on a set of data. The dashboard will allow the user to access visual representations of data.

*DataFilter*

The filter should be applied to data collected from external sources through external APIs, as well as the

*Analysis*

The algorithm will make use of tags and demographics set by the user. The algorithm must provide non-trivial analysis based on trends that may not be prevalent to the user. The algorithm must provide various perspectives with which the user can view the data in the ‘Graphical Results’ and ‘Dashboard’ contexts.

*PredictTrend*

The algorithm will be built adaptively in order to use deep-learning techniques to predict a trend after analysis.

*In-house DB*

The database will store user credentials, project information, analysis parameters and the results of running the algorithm. The database will not store graphical results built in the ‘Graphical Results’ context in order to save space and time, as an optimization trade-off. The charts and visual representations of analytics will be built in real time with the idea that users will be editing the points of focus in this analysis for each project multiple time.

*External API*

Tags will be searched for throughout widely-used Social Media platforms such as Twitter, Facebook and Instagram through API integration. User will then be provided with demographics they can use to organize data, such as age, location, density. Each query resulting from specification of a tag or demographic can be added to iteratively build a customized data set for analysis.

## Component Interface Identification

There are four main component interfaces:

* Frontend View
* Local Server
* Local Database
* Controller

## Software Component Concept of Execution

System Launch: The servers will check the connection to the app database and load the frontend interface. If any connection error occur the user will be notified. Common errors occur if the user performs directory change, externally modifies server data.

Authentication: The user is required to provide valid credentials or register. The given data is authenticated by the backend and if valid input is provided the user will be redirected to the dashboard. If the data is not corresponding to the initial server data, an error is raised.

User Local Data: The local data provided by the user will be encrypted and uploaded to the servers. Since the user is uploading the data externally the data will be checked from the servers for malicious content.

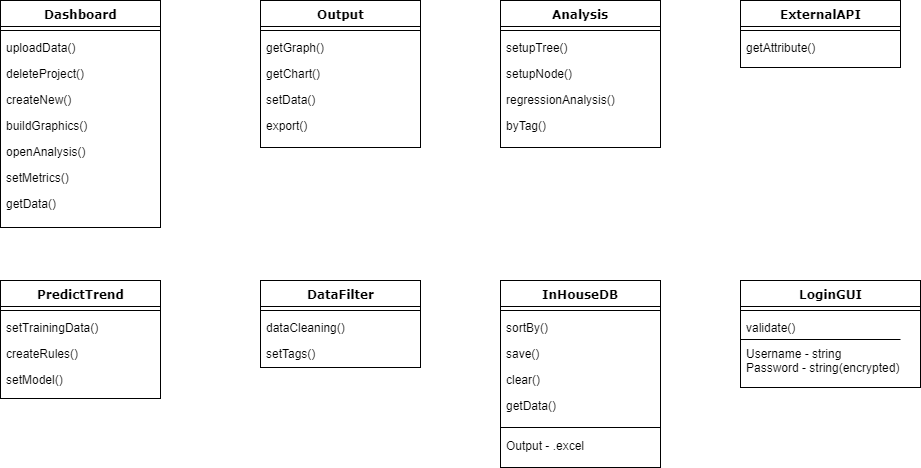
Results: The data will be analysed by the backend algorithm and the backend will request access for data modification. Then, the content will be displayed on the frontend/dashboard.

# SOFTWARE ITEM DETAILED DESIGN

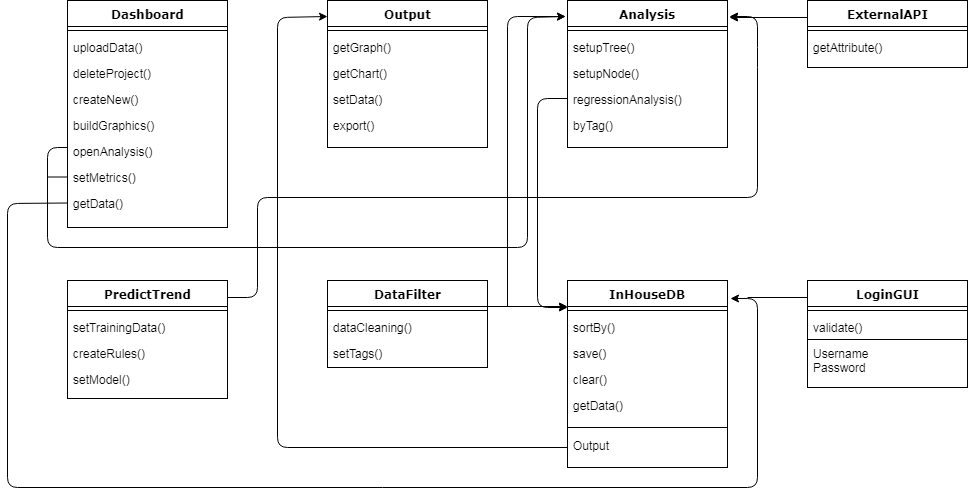
## Structure

### Software Unit Detailed Design

### 



## Static Relationship of Software Unit



### Run-time Object Instances

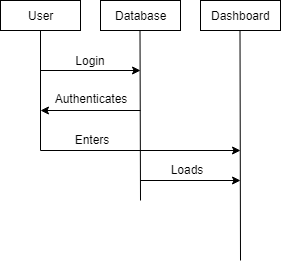
An instance is created when the Local Server makes a query request to either Local Database or via API to Social Network Database. Each query request is queued using FIFO queue. At the prototype stage of the application, server operations will not be processed in parallel, however, cache instances will be created in order to expedite similar queries made by other users.

## Behavior

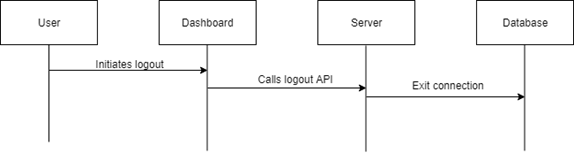
1. **Login**
2. User enters username and password
3. System authenticates credentials
4. If fails, system prompts user to retry. If users fail too many times, system will trigger mechanisms to ensure user is not bot (captcha for example)
5. Users can reset credentials in case forgotten.
6. **Dashboard**
7. Users can manage projects
   1. Open existing projects
   2. Create new projects
   3. Delete existing projects
8. Users can manage data to be used for analysis
   1. Upload custom data sets
   2. Create new data sets
   3. Organize and edit data sets
9. Users can access options to configure UI, output results, etc.
10. Users can access visual representations of data
11. **Social Networks Tool**
12. Users can search for tags and parameters to build a data set
13. Users can search through demographics of social network for data sets
14. Save data sets
15. **Graphical Results**
16. Users can export results to other file types
17. Users can save results
18. **Data Analysis Algorithm**
19. Users’ data sets are processed
20. If fails, prompts users to adjust data sets
21. Outputs results and predict trend
22. **Database**
23. Database is used for storing user credentials, projects, analysis parameters and results
24. System admin can access database to edit information inside database

### Sequence Interaction Diagrams

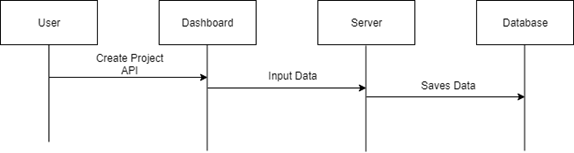
* Log in



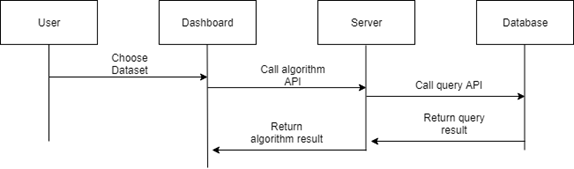
* Log out



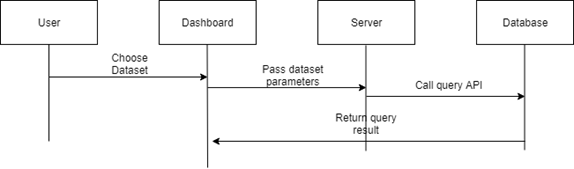
* Create Project API

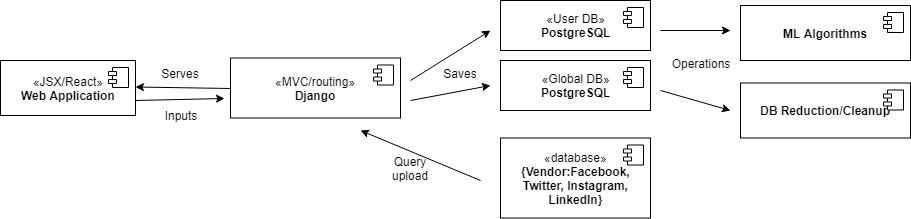


* Run Analysis



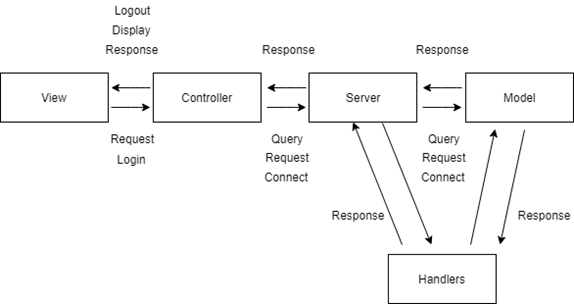
* Display Result





### Collaboration Diagrams

### C:\Users\Marc Tse\Downloads\Untitled Diagram (1).png

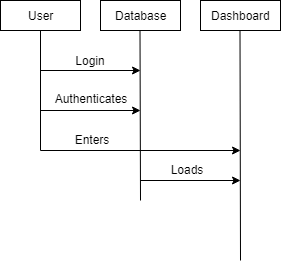


### Activity Diagrams

### https://lh3.googleusercontent.com/9ET0oVodIkUoU2JXIUPk6ir9rB_0WUTozx_IFG-R3kol9MaKEPQnekla-3w_8dLgSPXDDVshog_WDzaSBPXc5MvRv2I-Vhvh_j28fcxZ4gKd55jW2rPcGARcFRrgwDOIFXdRez4S

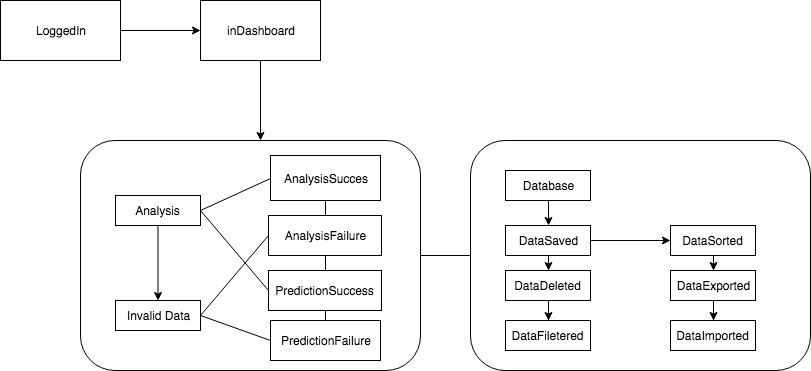
### https://lh5.googleusercontent.com/z0ACMuXkgReo89-iAUtG2IRJLZdLYmVxOozrgGO3AHCtf2HXa1pRwL9Ad0ybXNaFFUnGzGp13nlFuctJksEMXRizRpZL7KEZ3O-bhnpn5rVd7sB5I0yndi2hkmEe9fCMF-qbL2rbhttps://lh3.googleusercontent.com/ya0RqfM_BzhGxVEnDtAi4FLurQnWB--6OTR_Z0jWQNSSemoTQNpu0nxOM1ku6xS0U2ntTscRHydgweQnJDq8ptsiSYiVVncq0Nd4_-J7Nsvsu4UdG9bgDoLV9LAhGrRgQ546UNDkhttps://lh6.googleusercontent.com/8eME2oFIxXkTFv_SMekbMKX77IqfT4dIMzvjmYxoQDk_QjUK3YENbi2TMGTPRJ7I4uNZdYnMJbCiBXgCL2Gbacs0o-DtHeQ0gbuT8DT122Ix-N3pGhzaGzrdHMAE32rpqB0Q8McXhttps://lh3.googleusercontent.com/_R61IAIA1w3qHf4IKHjh9BvlREicXl5VOofJQjnqoAgzGZv_pZx92fy-ieyR34QOzkbSpdpRiSk0RcCK8xkhpeqBv2n9rduWRORlmKwr73ocMkEoHIUfHGDz6AQMY3SxseEG0BRnhttps://lh3.googleusercontent.com/ExPOeZ-0uOMYK8Y2r7WN9N1637nxjoiFEUeQ_8DyDGDKgEoLUMv4FV-e-xajVjl8C_o_2jesYt1vk9rCurtGr1DXBZqPCB_mwQrGlrPyGDHNg54edVNmtlOHe_GvwBLE3CHcuOsuhttps://lh5.googleusercontent.com/5V9J838vZva4vtONmUB-2c_mEXGzxn-4muWQMUWwLhJ184lqLlOjSaNWnSjw1a9vCoNWhLlM5fmXYUjpq9YywUu4xxfIYbruw5TxlKqTIaBjrvWCtwMiaRw4H6RniYRA5gk3LAS-

### State Diagrams

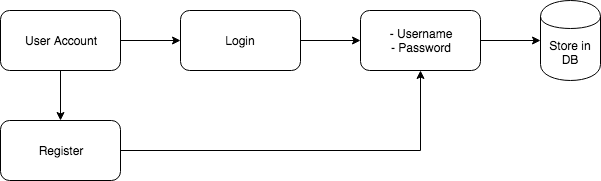


### C:\Users\Marc\Downloads\Untitled Diagram (13).png

### Event Diagrams



* Login/Register



* Data Import/Analysis



## Concept of Execution

When the application is open, the user must register to gain access to the server database. Then, the user must log in. If the credentials used are not corresponding to the initially registered information, an error will be raised. After the log in process is successful the user is directed to the dashboard, where his account information will be available. Graphs for user analysis and history of user’s interaction are also displayed. In other words the dashboard contains all the information and functions needed to use the app. If a user does not use the app for a day, the account is automatically logged out and directed to home page.

## Interface Design

Web Server controls the templating of the front end of application. User interacts with controls on the frontend, and Web Server will send user controls to the backend of application. Web Server also renders dynamically data sent from backend to the front end through a virtual DOM.

Local Server controls the logic of the application. Data sent from frontend is processed in the server, and corresponding actions, such as machine learning algorithms, or routing of the web application will be performed on the server. If CRUD operations are needed to be performed on the local database, Local Server will send RESTful calls to the database.

Local Database stores all information regarding individual user accounts, as well as global data such as machine learning optimization parameters suitable for all machine learning operations. Local Database performs queries as requested by the Local Server through RESTful calls.

Social Network Database is an external blackbox database operated by external entities. Social Network Database performs queries as requested by the Local Server through vendor APIs, and data will be returned to the Local Server for operations.

### Unique identifier of Interface

Each interface has a string UUID.

Web Server - 748244f2-c036-4c4c-9916-6789255be2ad

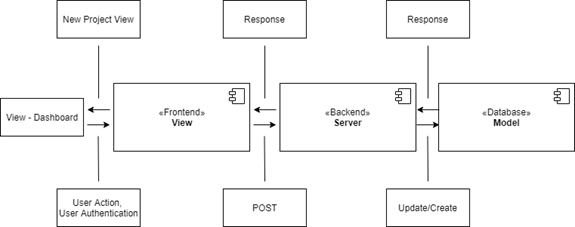
Local Server - b263b51c-0567-4222-a920-7674a53117ea

Local Database - 6503c496-c763-4c8b-b257-9b30b9140f6e

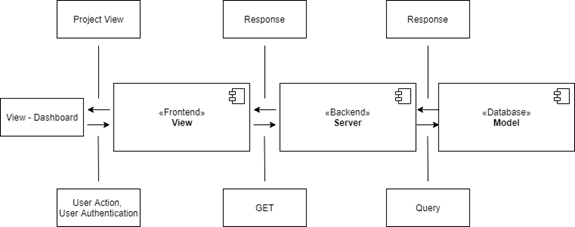
Social Network Database - 68936df0-b272-4fa5-8e33-ed489aeb6452

### Interface Diagrams

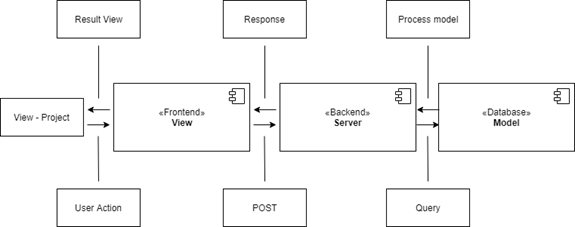
* Import/Create Project



* View Project



* Run Analysis



# IMPLEMENTATION ARCHITECTURE (NOT REQUIRED)

## All Active and Passive Classes Assigned to Components

*To be included in future release of this document.*

## Diagrams of Physical Packaging of Logical Components

*To be included in future release of this document.*

# DEPLOYMENT ARCHITECTURE

## Physical Deployment Architecture Diagram

## C:\Users\Marc Tse\Downloads\Untitled Diagram.png

# DICTIONARIES

*Refer to section 13.1 in Appendices for Dictionaries.*

# SOFTWARE ITEM COMPUTER RESOURCE UTILIZATION

Each software item should be optimized to occupy computer resources at the minimum.

React web application – RAM usage: <120Mb

Local server - <1GB

Local DB - <1GB/user

Social Network Database – R/W speed: 150 queries/min

# REQUIREMENTS TRACEABILITY

## Software Component-Level Requirements Traceability

All requirements will be traced using the traceability matrix. The design of each test case will strictly adhere to the thorough testing of requirements in order to satisfy forward traceability, while each test case conducted and its subsequent satisfaction of requirement will be recorded to track the progress for the completion of requirements as per backward traceability.

All documentation created during the project development can be traced from the initial source. In the first release (Version 1.0) of the Requirements and Analysis Specification (RAS), the source for traceability was the project proposal. But with every modified version of the RAS document, more sources of traceability is provided. These release changes will be indicated in Section 2.1.

# SYSTEM DESIGN TESTING

Verification and validation will be done with unit and acceptance tests. Verification is done between team members through daily SCRUM meetings, followed by formal weekly meetings. Prototyping of application and modeling of algorithm used in application will be heavily used for the verification process of development. At the beginning of each iteration of development of application, a prototype of the core priority of development should be created in order to verify the feasibility of the application. The prototyping process should not take longer than twelve hours, and in extreme cases where development is lagging behind schedule, prototyping process should not be used. Instead, a less labor-intensive method such as traceability of metrics is applied. Our customers will help in the validation process with validating if requirements and tasks are being satisfied. This can be done in weekly formal meetings, which will include demonstrations of baseline iteration of core function of product and project status.

Testing of the validity of the results of the algorithm will be done using historical data in order to verify analysis results. This will be the basis of testability in which previous marketing or advertisement campaigns will be reviewed holistically ad algorithmically with the results compared. Other features of the SNADL, such as the graphics building section can be reviewed similarly by verifying against historical data. The remaining features require simple testability, in that they can be easily validated by test use of their functionalities, such as the GUI.

## SQA Processes

# 10.1.1 Features for testing

- Log in / Log out

- User data import

- Local Database privacy

- User data modification

- Data Accuracy

# 10.1.2 Testing Approach

Several testing account will be created on the server to test each feature separately. Then the data from all account will be gathered for debugging. Testing will be done for reliability, mobility, precision, reusability and flexibility.

# 10.1.3 Results

If any unknown defects occur, the project will be analysed carefully for debugging. Any minor errors and bugs, that do not affect the user experience, will be considered as passing only for Beta-testing. For a full release, all components should be working without any warnings and bugs in order to provide full experience for the end user.

# RATIONALE

*None*

# NOTES

*None*

# APPENDICES

## Dictionaries

Classes

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Methods | Attributes |
| Dashboard | GUI of Dashbaord | uploadData() |  |
| deleteProject() |
| createNew() |
| buildGraphics() |
| openAnalysis() |
| setMetrics() |
| getData() |
| LoginGUI | GUI of Login | validate() | Username |
| Password |
| Analysis | Algorithm for data anlysis | setupTree() |  |
| setupNode() |
| regressionAnalysis() |
| byTag() |
| ExternalAPI | APIs of social networks | getAttributes() |  |
| PredictTrend | Algorithm for prediction | setTrainingData() |  |
| createRules() |
| setModel() |
| InHouseDB | Database | sortBy() | Data |
| save() |
| clear() |
| getData() |
| Output | Result of algorithm | getGraph() |  |
| getChart() |
| setData() |
| export() |
| DataFilter | Algorithm for managing data | dataCleaning() |  |
| setTags() |

Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Class | Arguments |
| uploadData | Upload custom data | Dashboard | Data |
| deleteProject | Delete existing project |  |
| createNew | Create new project |  |
| buildGraphics | Build graphics from output | Output |
| openAnalysis | Transition to Analysis |  |
| setMetrics | Set Data metrics |  |
| getData | Get data metrics | Data Metrics |
| validate | Validate user credentials | LoginGUI | Username |
| Password |
| setupTree | Set up tree data structure | Analysis | Data Metrics |
| setupNode | Set up node data structure | Data Metrics |
| regressionAnalysis | Analyze by regression | Data Metrics |
| byTag | Analyze by tag | Tags |
| getAttributes | Get data attributes | ExternalAPI |  |
| setTrainingData | Set training data for algo. | PredictTrend | Data Metrics |
| createRules | Create rules for algo. |  |
| setModel | Set up deep learning model |  |
| sortBy | Sort data | Database | Tags |
| save | Save data | Data |
| clear | Clear data | Data |
| getData | Get output data | Output | Output |
| getGraph | Get output graph | Output |
| getChart | Get output chart | Output |
| setData | Set output data | Output |
| export | Export to other format | Output |
| dataCleaning | Trim data | DataFilter | Data |
| setTags | Set data tags | Tags |

Attributes

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Description | C/S | Type | Size | Attribute | Attribute | Attribute |  | R/W |
| Username | Username to login | S | string |  |  |  |  |  | R,W |
| Password | Password to login | S | string |  |  |  |  |  | R,W |
| Output | Output of algo. | C |  |  | Data |  |  |  | R |
| Data Metrics | Used for data analysis | C |  |  | Data |  |  |  | R,W |
| Tags | Used to set rules for algo. | C |  |  | Data |  |  |  | R,W |
| Data | Generic container of data | C |  |  | Strings | Int | Double |  | R,W |

Relationship

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Description | From class | To class | Optional/Mandatory | Cardinality |
| Imports To | Data from API imports to analysis | ExternalAPI | Analysis | Optional | Many to 1 |
| Open | Go from one class to another | Dashboard | Analysis | Mandatory | 1 to 1 |
| LoginGUI | Dashboard | Mandatory | 1 to 1 |
| Validates | Validate credentials of LoginGUI with data in InHouseDB | InHouseDB | LoginGUI | Mandatory | 1 to 1 |
| Generate | Run analysis and generate outputs | Analysis | Output | Mandatory | 1 to Many |

Key Events

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Description | Motive | Action | Pre-conditions | Post-conditions | State Changes |
| LoggedIn | User is logged in | To use system | Log in thrugh username and password | User Credential validated | Transition to dashboard | From login screen |
| InDashboard | User is in dashboard | To access system's features | Create, open, delete project | Is logged in | Transition to analysis | From loggedIn |
| AnalyzeSuccess | Data analysis is successful | To analyze data | Input data for analysis, and click "analyze" | Valid data is inputted | Transition to data prediction | From dashboard |
| AnalyzeFailure | Data analysis is not successful | To analyze data | Input data for analysis, and click "analyze" | Invalid data is inputted | Prompts user to re-enter data | From dashboard |
| PredictSuccess | Data prediction is successful | To predict trend | Input data for analysis, and click "analyze" | Valid data is inputted, transitioned from analysis | Transition to output | From dashboard |
| PredictFailure | Data prediction is not successful | To predict trend | Input data for analysis, and click "analyze" | Data metrics is not sufficient, transitioned from analysis | Alert user of prediction failure | From dashboard |
| DataSaved | Data is saved to database | Prevent data loss | System automatically saves results to database or user edit database manually | Data is valid for saving | Alert user of success of action | From data prediction |
| DataDeleted | Data is deleted from database | Remove data | User selects which data to delete | Data is in database | Alert user of success of action | From data prediction |
| DataExported | Data from database is exported to other formats | Convert data | User selects which data to be exported, as well as format of the exported data | Exported format can be supported | Alert user of success of action | From database |
| DataImported | Data from external APIs is imported to database | Use data from external sources | User imports data through API | Data is valid to be imported | Alert user of success of action | From APIs |

## UML diagrams

## *None*

## Schedule Tracking

Time (hours)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| SPMP | Marc | 13 | 14 | -1 |
|  | Mehmed | 14 | 11.5 | 2.5 |
|  | Team summary | 27 | 25.5 | 1.5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| RAS | Marc | 23 | 21.5 | 1.5 |
|  | Mehmed | 28.5 | 21.5 | 7 |
|  | Team summary | 51.5 | 43 | 8.5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| SDD | Marc | 15 | 13.5 | 1.5 |
|  | Mehmed | 16 | 12.5 | 3.5 |
|  | Team summary | 31 | 26 | 5 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual and Team) | Estimated | Actual | Difference |
| Marc | 51 | 49 | 2 |
| Mehmed | 58.5 | 45.5 | 13 |
| Team summary | 109.5 | 94.5 | 15 |

## Defect Tracking

Counts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| SPMP | Marc | 5 | 2 | 3 |
|  | Mehmed | 9 | 2 | 7 |
|  | Team summary (Avg.) | 7 | 2 | 5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| RAS | Marc | 36 | 30 | 6 |
|  | Mehmed | 25 | 30 | -5 |
|  | Team summary (Avg.) | 31 | 30 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artifact or Deliverable | Who (individual and team) | Estimated | Actual | Difference |
| SDD | Marc | 20 | 12 | 8 |
|  | Mehmed | 12 | 12 | 0 |
|  | Team summary (Avg.) | 16 | 12 | 4 |

**Cumulative**

|  |  |  |  |
| --- | --- | --- | --- |
| Who (individual and team) | Estimated | Actual | Difference |
| Marc | 61 | Not complete | Not complete |
| Mehmed | 46 | Not complete | Not complete |
| Team summary (Avg.) | 54 | Not complete | Not complete |

