**Abstract**

The report provides details on the reverse engineering performed in order to derive the concrete architecture from the source code. The concrete architecture was compared with the conceptual architecture from the first part of the project. The rational for any architectural drift and erosion is mentioned. Any significant changes in the detailed design are also included along with appropriate explanation and rational for the differences.

**1. Introduction**

The purpose of the report is to highlight the software architecture recovery process performed by looking at the source code. Additionally, the report emphasizes any detected deviations from the conceptual architecture detailed in the first part of the project. The scope of the report is limited to extracting the concrete architecture of the main subsystems and comparing this to the conceptual architecture of the subsystems. The most significant changes in the detailed design and also identified and elaborated upon.

**2. Software Architecture Recovery Process**

TODO: Description of the process and tools used to recover the software architecture from the implementation artifacts, such as source code and script files

Also include the listing of all the types of code relationships, such as called-by, used to infer higher-level dependencies

**3. Description of the Architectural Drift and Erosion**

[Insert explain the rationale for the architectural drift (changes that are not in conflict with the conceptual architecture) and erosion for each diagram with respect to conceptual diagram in assignment 1]

First Level Decomposition (Relation between Subsystems)



Figure 1: High level Subsystem Decomposition

Second Level Decomposition (Subsystems and Components)

**Client Side Subsystems**



Figure 2: User Interface subsystem

**Server Side Subsystems**



Figure 3: Security Management subsystem



Figure 4: Database Management subsystem



Figure 5: Social Media subsystem



Figure 6: Food Log subsystem



Figure 7: Login Management subsystem

Third Level Decomposition (Classes with Significant Drift and Erosion)

[insert all class diagrams highlighting changes with respect to change in interacting with DB]

The most significant changes observed in the process of extracting the concrete architecture were in database management subsystem. This was so because initially we assumed that the Food Log, Social Media and Login Management subsystems would have their own database management classes. However, during implementation we used a common class named ‘DatabaseConnectionServiceImpl’ because all the other subsystems use the same interface (‘DBConnectionServiceAsync’ and ‘DBConnectionService’) to make RPC calls to the database. This is the rationale for this major difference in many of our class diagrams.