|  |  |
| --- | --- |
| Category | Details |
| Program Organization | * Main App   + Login   + List conversations   + Analysis chooser option   + Message Downloads   + Options Menu * Separate analysis components   + Option ui   + Message receiving and analysis (parallel option)   + Separate databases |
| Major Classes | * UI   + Welcome/Conversation List   + Authentication   + Downloading Messages Dialog   + Analysis Options (checkboxes)   + Custom UI’s for modules (e.g. printed tables)   + Options     - Custom User Controls for different analysis modules * Data Management   + Class to interact with Facebook: FBInteractor * Interfaces   + AnalysisModule |
| Data Design | * Message Data (Schema)   + Tables: 1 per conversation     - Column: Message String (can be null)     - Column: Date NOT NULL     - Column: Sender NOT NULL * List of Analysis Modules * Analysis Module Specific (Separate Schemas)   + Tables: Dependent on modules (e.g. swear words/users/etc.) * Options (Schema)   + Tables: 1 per analysis module that uses options/preferences     - Columns: defined by the analysis module |
| Business Rules | * Read all of the messages from the server * Analyze messages using message package * Allow logging on and off for users * Allow users to select which group/conversation to analyze |
| User Interface Design | * Welcome Screen/Conversation id sifter * Authentication popup * Downloading Messages popup * Select Analysis Menu * Options/Preferences Menu * Analyzing popup * Analysis Result popup (analysis module specific) |
| Resource Management |  |
| Security | * Way to lock database? * Allow only read message access for Facebook * FB interface will prevent any malicious activity against FB profile |
| Performance | * Goal: 500 messages/second – approx. 3min 20 seconds for 100,000 messages |
| Scalability | * Expandable * Design must handle large volume of messages |
| Interoperability |  |
| Internationalization /  Localization |  |
| Input/Output | Input:   * FB messages * User FB authentication   Output: |
| Error Processing |  |
| Fault Tolerance |  |
| Architectural Feasibility |  |
| Overengineering |  |
| Buy vs. Build | Buy:   * JSON analyzers   Build:   * Database * UI * Options/Preferences interfacing |
| Change Strategy |  |

Architectural Feature Not Being Considered:

1. Overengineering
2. Change Strategy