Tetranucleotide Frequency Viewer SRS

Version 1.0

Aaron Chamberlain - *Project Manager*

Adrian Osuna - *Assistant Project Manager*

James Mosley - *Software Engineer (Gestures)*

Juan Morales - *Software Engineer (Graphing)*

Justin Camarena - *Software Engineer (File I/O)*

Javier Zarate - *Software Engineer (Layer Views)*

Jocelyn Mireles - *UI/UX Designer*

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE

The purpose of this worksheet is to specifically define the software requirements for the Tetranucleotide Frequency Viewer mobile app being developed by the CSE455 Software Engineering team at CSUSB. The mobile application is being developed for use by Dr. Jeremy Dodsworth of the Biology Department at CSUSB, as well as other faculty.

This team will satisfy Dr. Dodsworth's requirements to facilitate the applications use in his research and teaching. Our goal is to have a functional prototype that can be improved on after the end of the 10-week time frame of the course.

1.2 SCOPE

This mobile application is being designed specifically for the Android operating system and Tablet devices. The application will have only one main view, with 3 graphical overlays that each control specific functionality of the application. The main view or 'Tetranucleotide Frequency Graph' will be where all data will be plotted in 3D space. The first overlay will be for 'File & Layer Selection' where the user will select the files they would like to display, once the files have loaded the user can toggle layers on and off to simplify the graph. The second overlay will be for 'Graph Animation', where the user can select one or more axes about which to rotate the graph, along with a slider to control the rate at which the camera moves. Upon selecting a single point or a range of points, the user will see the third overlay or 'Detailed Information View'. This overlay will provide information about the selected data, such as: coordinates, genus, fragment number, and copy number. We are currently targeting Android launch. The app will pull all data from local files found on the device itself, whether it be internal storage or external such as a microSD card.

The data itself has been prepared by students in Bioinformatics and is made available as a CSV file. Future versions of this app should be able to take standard multi-FASTA format files and prepare the data file, whether it be locally or on a remote server. Due to the steep learning curve of OpenGL ES, the first prototype will be a 2-D graph, with the layer view fully implemented, as well as the additional information overlay and some gestures. The second prototype will make the graph 3-D, will add in the remaining gestures, and will add in the graph animation functions.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Android	Google's mobile operating system
Android Studio	Google's IDE for android development
OpenGL ES	Cross-platform API for 2D & 3D graphics rendering
IDE	Integrated Development Environment
3G	Third generation of wireless data standard
4G	Fourth generation of wireless standard, typically LTE or HSPA+
HTTPS	Secure transfer protocol for server communication
Java	The language used for Android development
WiFi	Wireless internet for devices
SRS	Software requirement specifications

SDK	Software development kit provided by a vendor
Multi-FASTA Format	A text file format for representing nucleotide sequences.
Tetranucleotide	A codon which contains four nucleotides.
Genus	A taxonomy term to denote different branches of a species.
Copy Number	The number of times a particular tetranucleotide segment was present in sequenced data.

1.4 REFERENCES

- Google Development
  http://developer.android.com/index.html
- ADA Section 508 Compliance
  https://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards
- OpenGL ES
  http://developer.android.com/guide/topics/graphics/opengl.html
- IEEE SRS Standard Template
  IEEE Software Requirements Specification Template

1.5 OVERVIEW

This document will cover the specifications and interfaces used by the mobile application along with its requirements as dictated by the design constraints. The next sections will show the user interface and describe where certain functions will be implemented in the system.

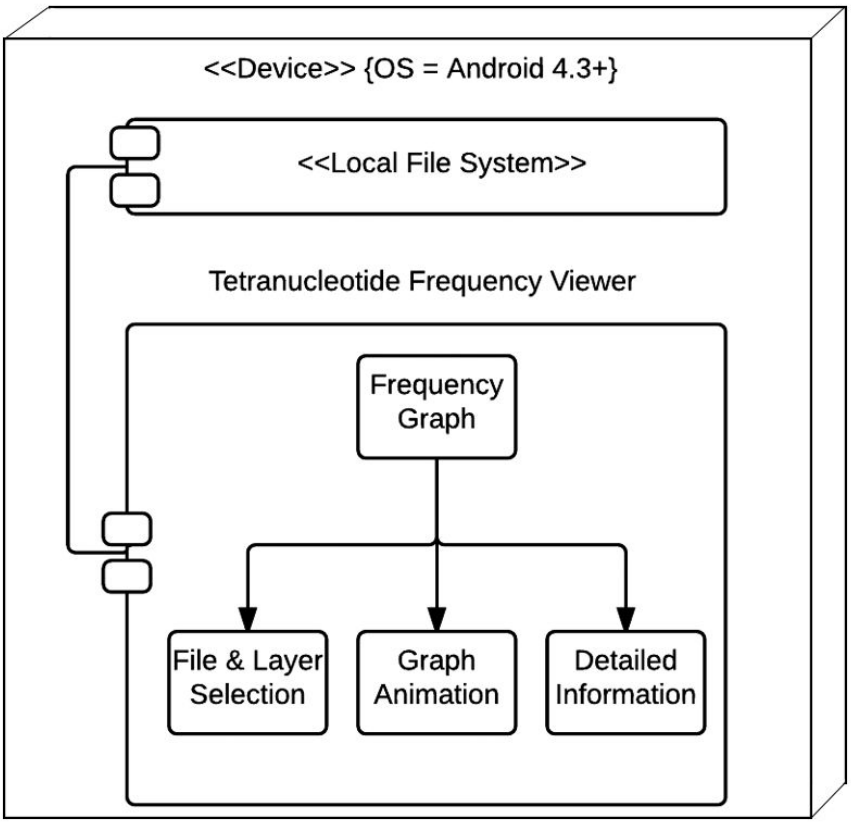
2 OVERALL DESCRIPTIONS

2.1 PRODUCT PERSPECTIVES

The purpose of this application is to facilitate biological research and learning through advanced graphing techniques. The graph will display the tetranucleotide frequencies of a given biological sample using Principle Component Analysis in 3D space, and allow for manipulation through several unique tools.

This application will consist of the main graph view and three unique overlays which can be pulled up or hidden. The overlays will provide access to File & Layer Selection, Graph Animation, and Detailed Information. Currently, the application is being designed for Android Tablets, but given the portability of OpenGL ES to other Mobile Operating Systems, the code will be reusable and straightforward to bring to iOS or Windows Platforms.

2.1.1 SYSTEM INTERFACES (DEPLOYMENT DIAGRAM)



The application will be written in Java, using the OpenGL ES API for 3D viewing, and the Android SDK for all User Interface elements such as buttons and text fields. All genomic data files will be handled by the user outside the scope of the application, and will simply be opened by the application.

2.1.2 USER INTERFACES

- Tetranucleotide Frequency Graph (Main View) - This displays the frequency cluster graph of the selected genomes. The camera angle of the graph can be rotated and zoomed in.
- File & Layer Selection (Overlay) - Brought up by tapping on the left of the screen, allows the user to load files from their device to be graphed and to toggle layers on and off to simplify the view.
- Graph Animation (Overlay) - Brought up by tapping on the top of the screen, allows the user to continuously rotate the graph about any given axis and to control the speed of this movement.
- Detailed Information (Overlay) - Brought up by tapping an individual point or selecting several points. Relevant information such as coordinates, genus, fragment number, and copy number.

2.1.3 SOFTWARE INTERFACES

- OpenGL ES - Used to render all 3D & 2D graph elements.
- Java - Used to write all other app logic, including user interactions and call OpenGL
- Android Studio - The IDE used to development the application and simulate the app.

2.1.4 COMMUNICATION INTERFACES

The application doesn't require any form of communication such as WiFi or 4G, as all files are handled by the user outside the scope of the application. This also allows the user to present the data offline.

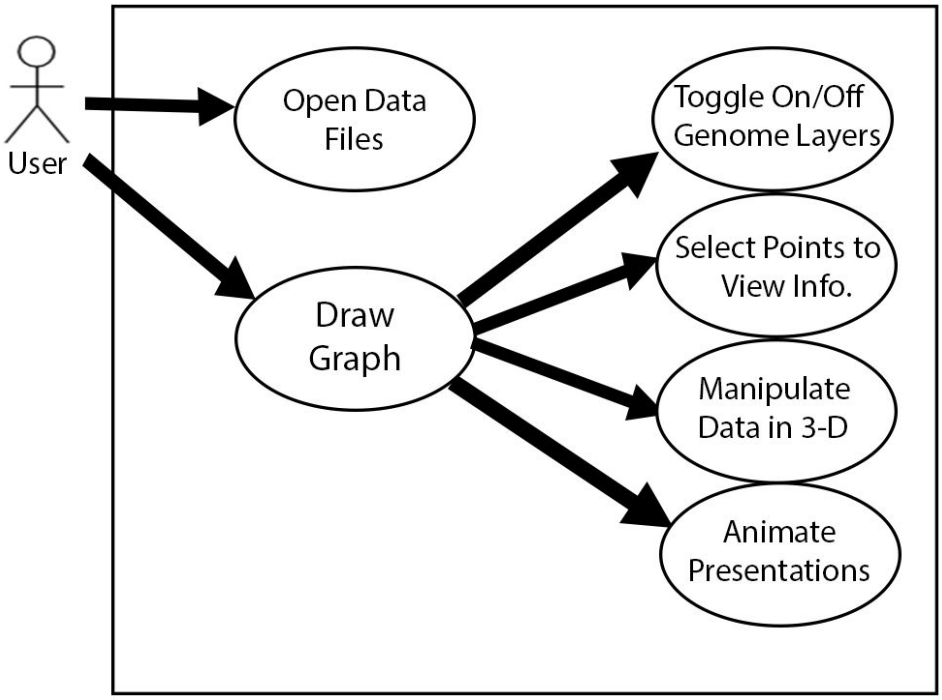
2.1.5 MEMORY

This application will be dependent on a device's capability, it will not load more data points than it will be able to hold in memory. We are targeting a minimum Android version of 4.3 (Jelly Bean) as this update includes OpenGl ES 3.0.

2.1.6 OPERATION

The application will not have any planned downtime as all new updates will be installed by the user at their discretion.

2.2 Product Functions (Use Case Diagram)



2.3 USER CHARACTERISTICS

A user can be an employee or student of CSUSB, as the application relies on proprietary file input. This limiting factor can only be negated by expanding the scope of the application to take standard multi-FASTA files and run the conversion algorithm in real-time.

2.4 CONSTRAINTS

We have two main constraints for this mobile application.

- Operating system constraint - The user must have Android 4.3 or higher.
- ADA constraint - The mobile application should comply with ADA guidelines for accessibility to all users.

2.5 ASSUMPTIONS & DEPENDENCIES

- The only dependency is OpenGL 3.0 which is supported by Android 4.3 and higher.

3 SPECIFIC REQUIREMENTS

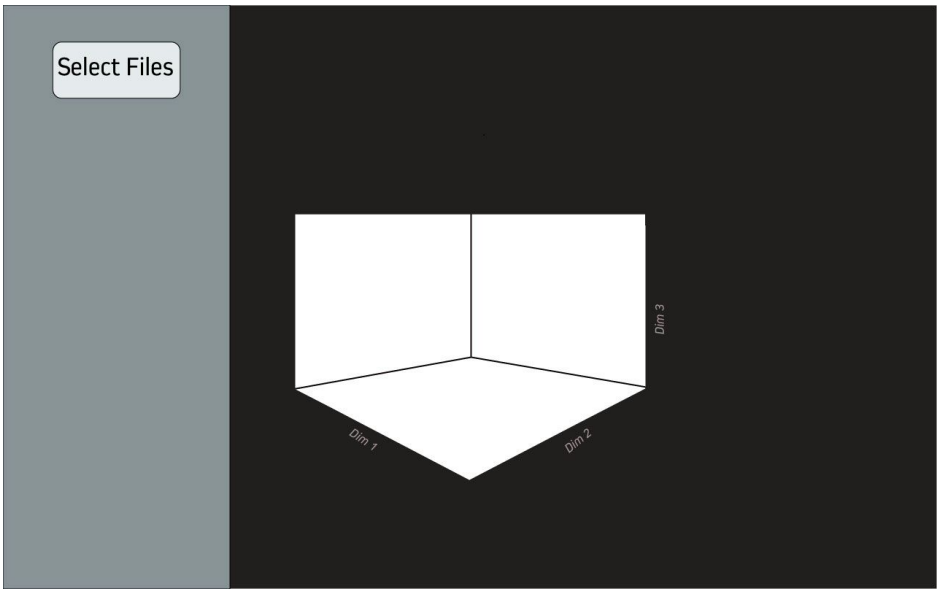
3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 USER INTERFACES

This section will show how the user interacts with the mobile application.

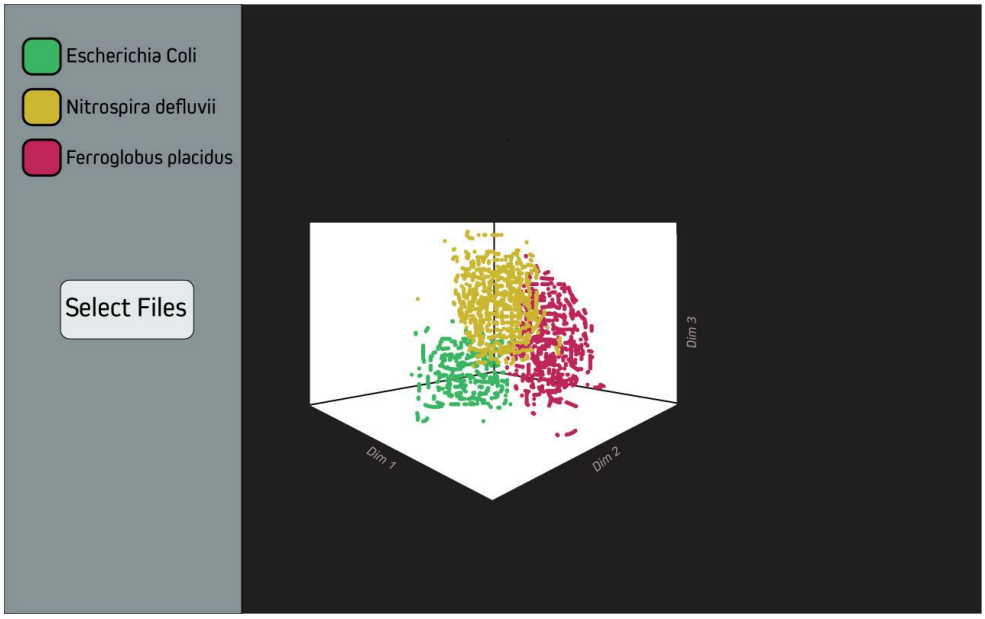
3.1.1.1 Tetranucleotide Frequency Graph/File Overlay

After the splash page, the user will see the bare axes of the graph and the menu to load their files.



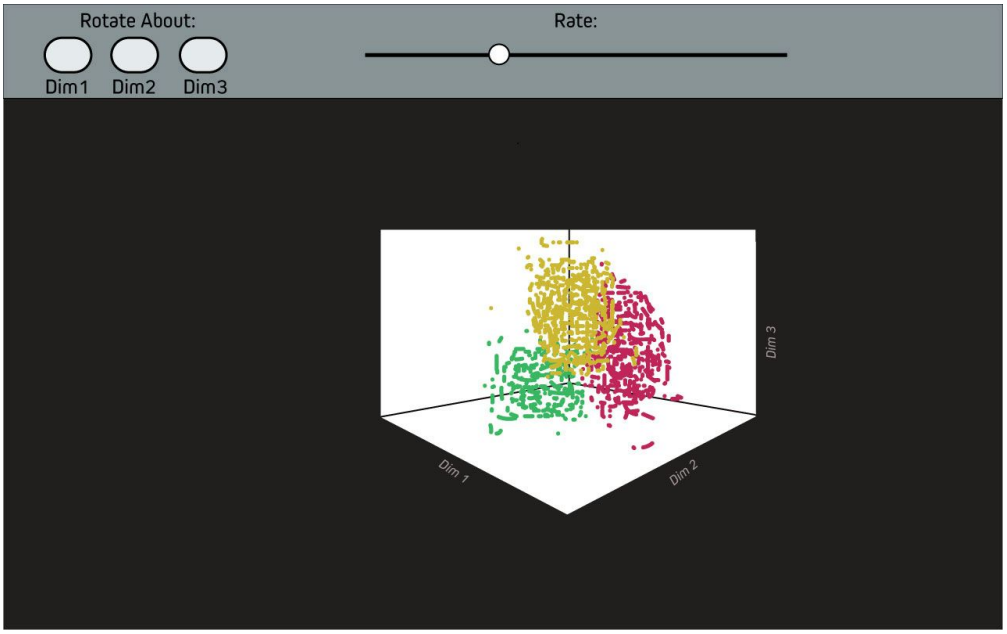
3.1.1.2 File & Layer Selection (Overlay)

After the files have loaded, the layer view will be populated, at which point the user can either add more files or begin to toggle on and off layers. All overlays are hidden again but tapping on the main canvas of the graph. The user can bring it up again by tapping the left side of the screen.



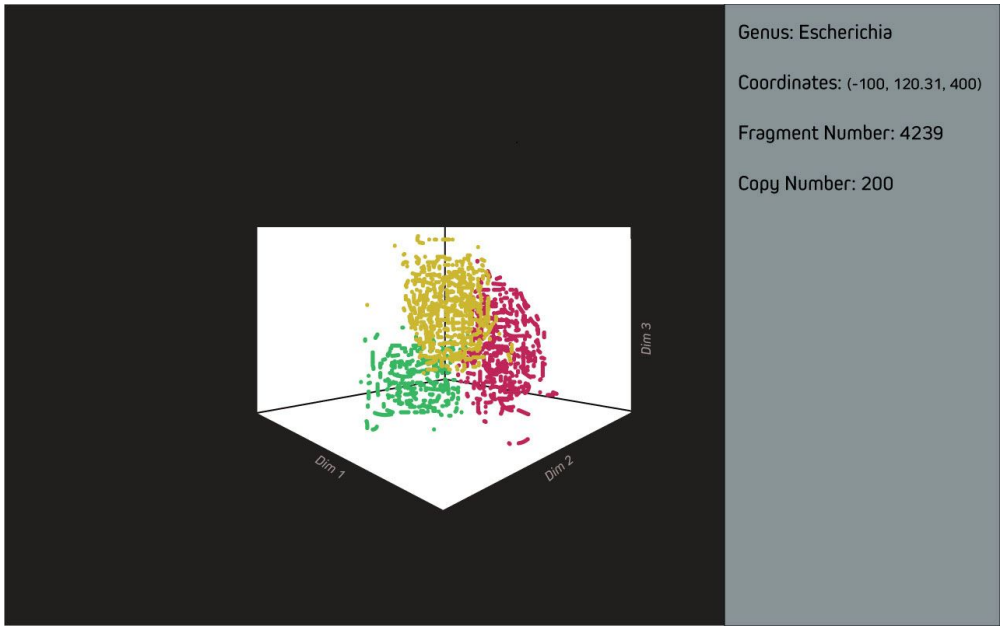
3.1.1.3 Graph Animation (Overlay)

The user brings this overlay up by tapping the top of the screen. It has three toggle buttons to select which axes the graph will rotate about. There is also a slider to control the rate of this movement. Lastly, it contains a reset button to get to the default camera angle.



3.1.1.4 Detailed Information View (Overlay)

The user brings this overlay up by either tapping on a single point on the graph or by selecting several points. Detailed information will be displayed here such as their coordinates, genus, fragment number, and copy number.



3.1.2 Hardware Interfaces

The application does not interface with any hardware device buttons, but may have access the internal storage and/or SD Card Slot. The application is a single instance application that is supported by Android 4.3 and higher.

3.1.3 Software Interfaces

The application is self-contained and does not access software outside of the application.

3.1.4 Communication Interfaces

The application reads files directly from the user's device, such as internal storage or SD card, and does not require an internet connection.

3.2 Functional REQUIREMENTS

3.2.1 Tetranucleotide Frequency Graph

3.2.1.1 Splash Screen

This will display the application logo and programmer credits until the main application view has loaded.

3.2.1.2 Graph View

Displays all graphed data points. Standard gestures may be used, such as pinch to zoom and one finger drag to rotate the camera angle.

3.2.2 File & Layer Selection (Overlay)

This overlay will allow the user to select the files they would like to display, and once rendered, each genus will be able to be toggled on or off.

3.2.3 Graph Animation (Overlay)

Contains 3 toggle buttons to select which axis the user would like to rotate the graph about. A slider will control the velocity of this movement, and a reset button will show the graph in the default camera view.

3.2.4 Detailed Information View (Overlay)

This overlay will contain information on a single graphed point, or a set of points such as: coordinates, genus, fragment number, and copy number.

3.3 Performance requirements

The application will have all art and non--database content locally on the user's mobile device. We aim to achieve Real-Time Rendering, or 30FPS to allow for smooth user interactions. At high data counts (10,000+ objects), temporary files will be used to allow for high performance and low RAM impact.

3.4 Design Constraints

The mobile application will adhere to the standards set forth by CSUSB, the ADA and the client's request.

3.5 Software System Attributes

3.5.1 Reliability

This application will be correctly formatted for the supported platforms and to be maintained to either access data quick and graph the data accordingly. The mean time between failure shall exceed 360 hours (15 days). The mean time to repair shall not exceed 12 hours (1 work-day).

3.5.2 Availability

Access to the application will be available around the clock, as all files are stored locally on the user's device.

3.5.3 Security

3.5.3.1 System

The application can be used completely offline, so the only security concern is verifying file input integrity before rendering them.

3.5.3.2 Data

All data will be handled through a server on the school network outside the scope of this application. This will allow the application to render the graphics rapidly. The input files will never be written to, but additional temporary and long-term files may be written to the user's device.

3.6 other requirements 3.6.1

TESTING

3.6.1.1 Unit Testing

Each view will be tested separately before being incorporated with the rest of the views to ensure functional **independence.**

3.6.1.2 Integration Testing

After unit testing, all the views will be integrated into the mobile application where they will be tested in different sequences.

3.6.1.3 Acceptance Testing

After unit and integration testing, this testing will be done on a device to ensure that the product is ready to be deployed to actual research activities.

3.7 DOCUMENT APPROVAL

Dr. Jeremy Dodsworth (Application Client):