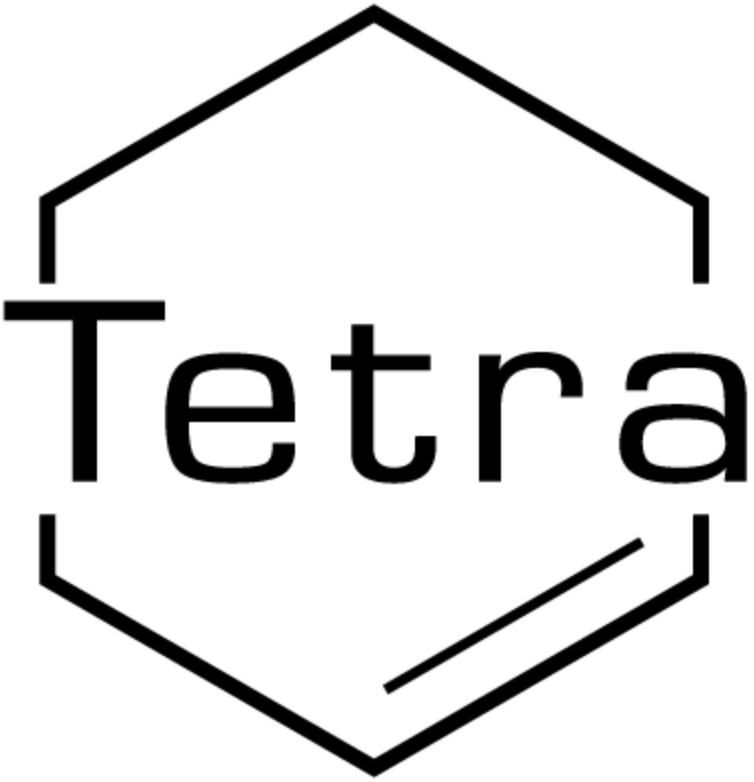
Tetra (Tetranucleotide Frequency Viewer)
SRS
Version 1.1



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, that will have a graphical display. The main view or 'Tetranucleotide Frequency Graph' will be where all data will be plotted in 3D space. There will be a side panel to "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. 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 WebGL, the first prototype will be a 2-D graph, as well as the additional information and some gestures. The second prototype will make the graph in 3-D but will not have all the functions that were available on prototype 1.

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.
JavaScript	High level interpreted programming language, used for web based applications.
WebGL	Web Graphics Library is a JavaScript API for rendering 3D and 2D graphics with any capable web browser.
three.js	Cross-browser JavaScript Library/API to create and display 3D graphics with the use of WebGL
Splash Screen	Display Logo at the start of the application.
PapaParse.js	Open source library used to open .CSV file and parse to JSON format using the W3 File API.

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
- Three.js
  http://threejs.org/
- Node.js
  https://nodejs.org/api/
- Papa Parse
  http://papaparse.com/

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.

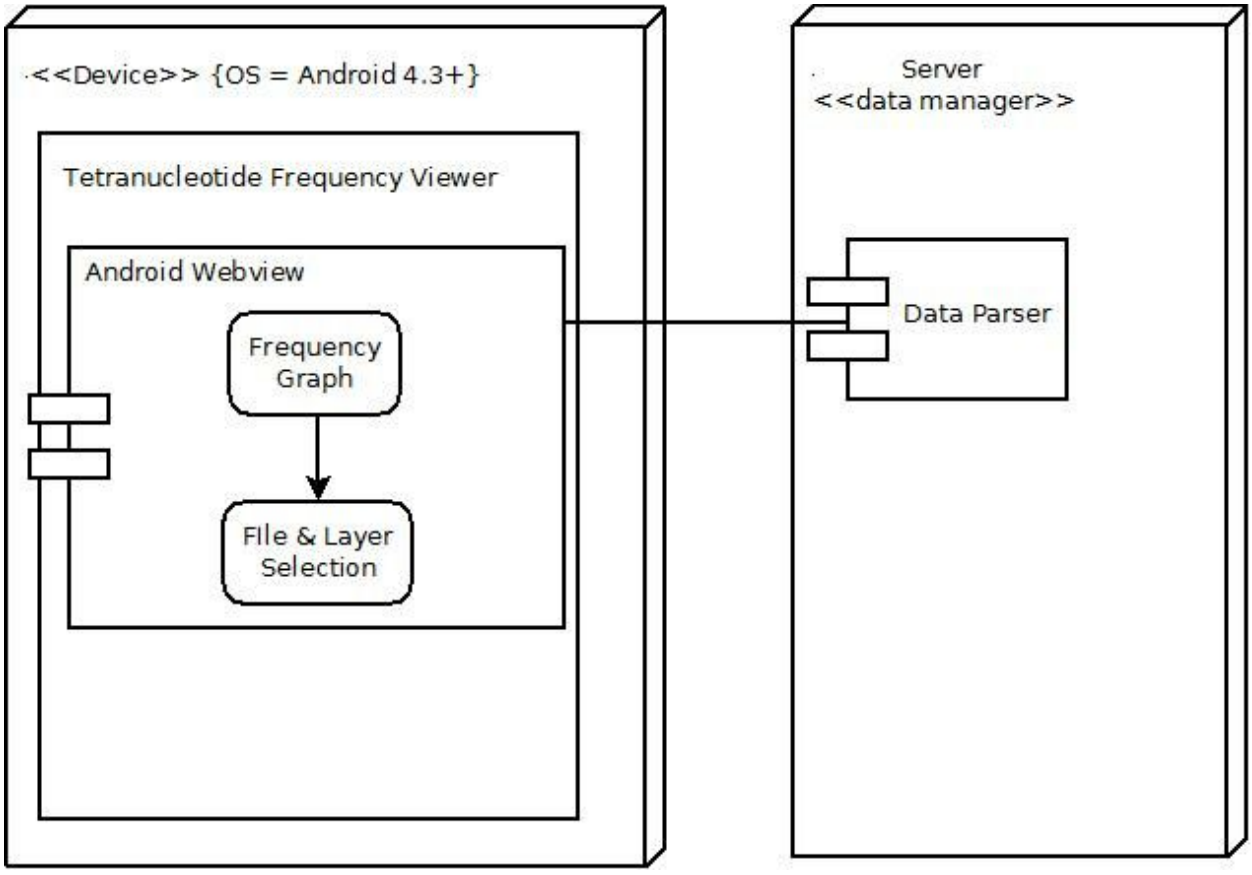
2 OVERALL DESCRIPTIONS

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 a single overlay which can be pulled up or hidden. The overlay will provide access to File & Layer Selection and Detail Information. Currently, the application is being designed for Android Tablets, but given the portability of Android's webview to other Mobile Operating Systems, and not limited to just a table but smaller devices such as a phone. The code will be reusable and straightforward to bring forth to iOS devices or Windows Platforms.

2.1.1 SYSTEM INTERFACES (DEPLOYMENT DIAGRAM)



The application will be written in Java, using WebGL API for 3D viewing, a nginx server will be used to serve up the static content, 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. The servers are stationed on the CSUSB campus, and will only work if the user is either connected through a VPN or on the CSUSB campus.

2.1.2 USER INTERFACES

- Tetranucleotide Frequency Graph (Main View) - This displays the frequency cluster graph of the selected genomes.
- File & Layer Selection - available along the left side of the screen, by tapping the button labeled 'Open File' it will toggle a new page allowing the user to load a file from their device to be graphed and to toggle layers on and off to simplify the view.

2.1.3 SOFTWARE INTERFACES

- HTML - Used to write the background display of the button's and graph overlay.
- CSS - Used to style and position the elements on the web page.
- WebGL - Used to render all 3D & 2D graph elements.
- Java - Used to write the android webview portion to call to the WebGL functions.
- JavaScript - Used to manage the files data and display the graph qualities.
- Android Studio - The IDE used to development the application and simulate the app.

2.1.4 COMMUNICATION INTERFACES

The application does request to access the user's WiFi or 4G connection, as all files are handled outside the scope of the application. This also allows the application to manage the data quickly and prevent large consumption of memory on the user's device.

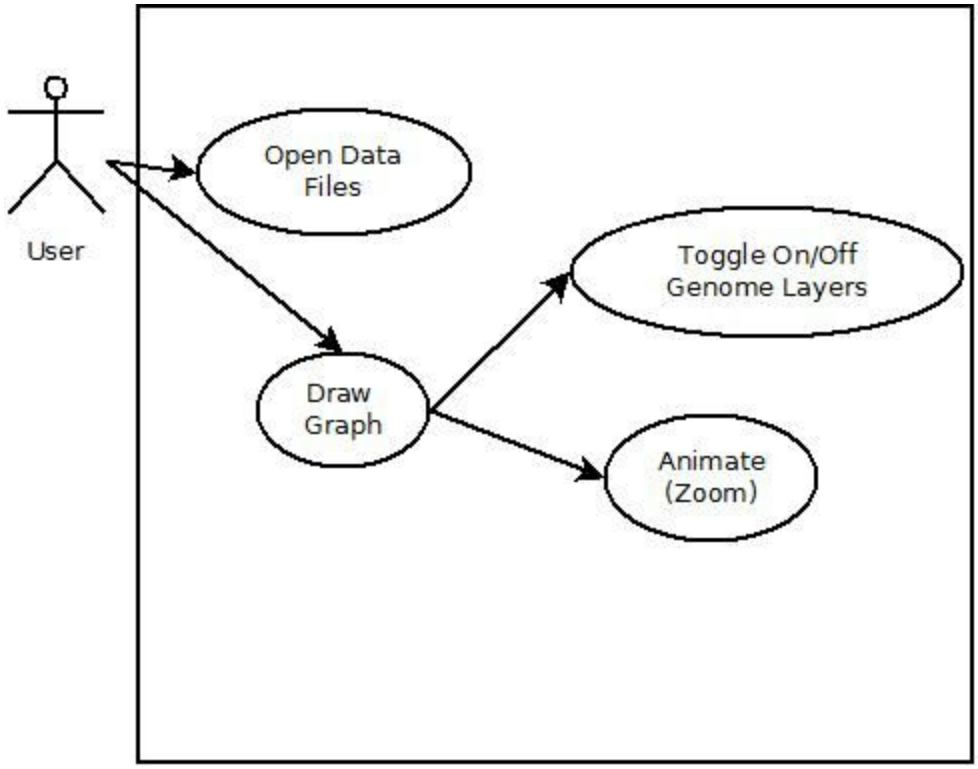
2.1.5 MEMORY

This application will be dependent on a device's capability to access the internet and the server. We are targeting a minimum Android version of 5.0 (Lollipop).

2.1.6 OPERATION

The application will have regular planned updates, as new code is pushed to the server. Any downtime from these updates will be in the magnitude of seconds as all server content is simply retrieved and then handled client side. The user is will simply receive these updates the next time they refresh the page or open the app again. Any server downtime will be at CSUSB IT department's 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 one main constraint for this mobile application.

- Operating system constraint - The user must have Android 5.0.x (Lollipop) or higher. WebGL did not become a Chrome standard until this point, and the W3 FILE API was also added in this update.

2.5 Assumptions & Dependencies

- The first dependency is on WebGL, which is partially implemented in Android 4.3 and 4.4, with full implementation in Android 5.0 and above.
- The W3 FILE API standard is also required by our third party library PapaParser. This standard has been fully implemented by most browsers as of 2014, and is incorporated into Android 5.0 and above.

3 Specific Requirements

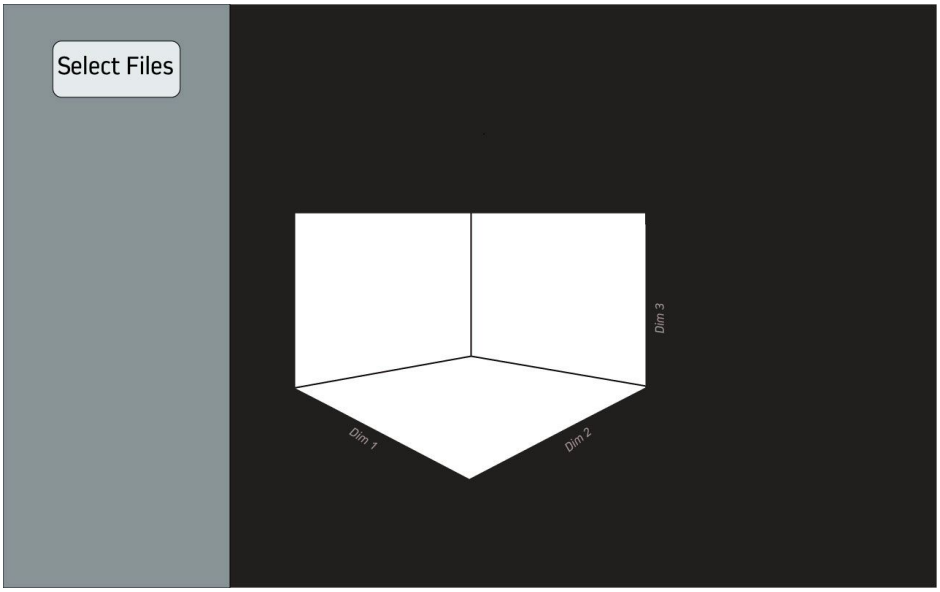
3.1 External Interface Requirements

3.1.1 USER INTERFACES

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

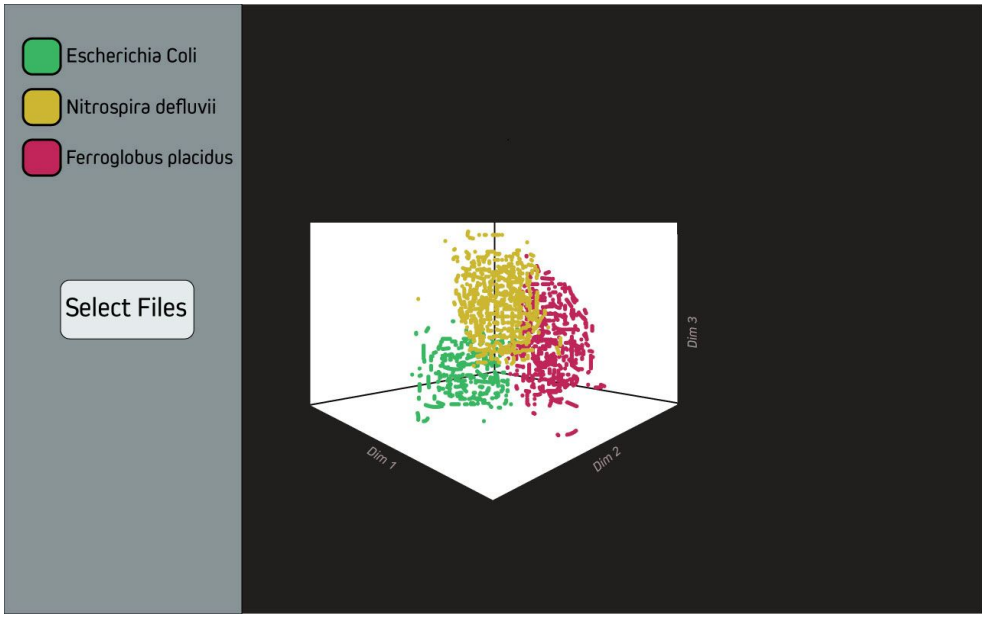
3.1.1.1 Tetranucleotide Frequency Graph/File Overlay

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

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.



3.1.2 Hardware Interfaces

The application does not interface with any hardware device buttons. The application is a single instance application that is supported by Android 5.0 and higher.

3.1.3 Software Interfaces

The application will require internet connection in order to access the remote content on the server.

3.1.4 Communication Interfaces

The application only needs to request the application files once, after this, all events are handled client side, including the opening and parsing of Files.

3.2 Functional REQUIREMENTS

3.2.1 Tetranucleotide Frequency Graph

3.2.1.1 Splash Screen

This will display the application logo 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.

3.2.2 File & Layer Selection

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.

3.3 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 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 workday).

3.5.2 Availability

Access to the application will be available around the clock. Even though the user must access a remote server for the data, the server will be online with exception of power outages and updates.

3.5.3 Security

3.5.3.1 System

The application only requests files from the server and does not send any data back, no login is required but the user must be on the campus network to access the server or be remotely connected through a VPN, so the only security concern is the loss of the data from the server to the device and 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 files may be created on the local device to maintain the expected performance.

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):