

Early Earthquake and Tsunami Warning Viewer

Yicheng Shao (Eason)

September 21, 2024


Abstract

Give a brief summary outline of your project.

©2024–2025 Yicheng Shao (Eason).

This work is licensed under a CC BY-NC-ND 4.0 license.

This work is formatted using Xe_{La}TeX. The source code is available at  EasonSYC/nea-report.

The relevant program source code is available at  EasonSYC/EEETWV, licensed under the MIT License.

Contents

1	Analysis	4
1.1	Background Information	4
1.1.1	The Early Earthquake Warning System	4
1.1.2	Earthquake Terminology	4
1.2	Problem Area	5
1.3	Client and End User	5
1.4	Research Methodology	5
1.4.1	Client Interview	5
1.4.2	Existing Applications and Solutions	7
1.5	Features of proposed solution	8
1.6	Critical Path	10
1.7	Requirements Specification	12
2	Design	15
2.1	Hierarchy Chart	15
2.2	Data Structures/Data modelling	15
2.2.1	External Data Sources	15
2.2.2	OOP Model	15
2.3	User Interface	15
2.4	Hardware Software Requirements	16
3	Technical Implementation	17
3.1	Key Code Segments	17
3.1.1	Data structures	17
3.1.2	Modularity	17
3.1.3	Defensive Programming/Robustness	17
4	Testing	18
4.1	Test Strategy	18
4.2	Testing Video	18
4.3	System Tests (against original requirements specification)	18
5	Evaluation	19
5.1	Requirements Specification Evaluation	19
5.2	Independent End-User Feedback	19
5.3	Improvements	19
6	Code Listing	21

1 Analysis

1.1 Background Information

1.1.1 The Early Earthquake Warning System

Earthquake is one of the most common natural disasters in the whole world, and direct consequences of earthquakes include tsunamies which could be catastrophic.

Japan, sitting on the intersection of the Eurasian, the Philippine and the North-American plates, is the countries with most earthquakes. Historically, the Great Kantō Earthquake in 1923, the Great East Japan Earthquake in 2011 (a.k.a. the Tōhoku Earthquake) and the recent 2024 Noto Peninsula Earthquake all caused hundreds of deaths, both due to the result of the earthquake(s) and the resulting tsunami.

To provide protection to its residents, the Japan Meteorological Agency (JMA), together with the National Research Institute for Earth Science and Disaster Resilience (NIED) placed thousands of **earthquake sensors** across Japan (the Hi-net), with several lying deep in the sea bed, measuring displacement, velocity and acceleration, which are connected to multiple servers, including two located in Ōsaka and Tōkyo.

Using data obtained from the sensors, computers do some complicated algorithms (mentioned below) to send out **early earthquake warnings (EEWs)** automatically within milliseconds. There are two types of EEWs:

1. **EEW (Forecast)**. Sent out to **highly-dependent industries** (e.g. rail industry, power plants) and **subscribed users**, when maximum intensity level of more than 3, or a magnitude of more than 3.5 is expected.
2. **EEW (Warning)**. Sent out to **everyone** via TV, Radio, Mobile Phone, SMS, etc., when a maximum intensity level of more than 4 is expected.

After the earthquake, JMA staff will determine the location and severity of tsunami warnings to be issued, if necessary.

1.1.2 Earthquake Terminology

- **Intensity**. The intensity describes the intensity vibration of a point due to an earthquake. It is not unique to an earthquake - **different places can have different intensities** due to the distance to the epicenter, and intensity will also change over time. JMA measures intensity using **9 levels: 1, 2, 3, 4, 5-, 5+, 6-, 6+ and 7** in increasing order.
- **Magnitude/Scale**. The magnitude of an earthquake describes the energy released in the earthquake in a logarithmic scale. **It is unique to an earthquake.**
- **Epicenter/Hypocenter**. The epicenter is the surface point directly above the true centre of the earthquake.
- **Focal Depth**. The focal depth is the depth of the true center of the earthquake.
- **P-Wave and S-Wave**. These are seismic waves, sourced from the true center of the earthquake, travelling at different speeds, with Primary (P)-Wave travelling faster and Secondary (S)-Wave travelling slower.

1.2 Problem Area

The main goal of this application is to provide a visualisation of the earthquake/tsunami related data feed(s) provided by JMA's affiliated institution, Disaster Mitigation Data Send Service (DM-D.S.S). There are numerous separate apps providing a list of recent earthquakes, the real-time data measured by the sensors, and the real time earthquake warning displayed on a map, but rarely are there good apps that combine all those features together in a satisfying way, with just the necessary features the author needs.

Some applications are no longer being updated due to change in the user's policy of the related data feed. Furthermore, most of the apps available are only in Japanese, not in English or my home language Chinese, which can create trouble for the author to understand.

1.3 Client and End User

The primary target of this application will be passionate geographers and geologists who are interested in the study of earthquake observations and predictions. The age group of this vary all the way from primary-school students to adults, including the author who has been amazed by the technology since the age of 12. They could take any employment, ranging from students to full-time jobs. Their proficiency usually varies, since there are people new to this field who probably does not have much knowledge, so the interface of the application should be relatively user-friendly and understandable, hiding unnecessary technical complexities.

Another target client could be industries which highly rely on earthquake predictions due to the risk imposed by earthquakes. High-speed railway and nuclear power plants are good examples of this. Therefore, the staff in charge monitoring will usually have higher proficiency and would like more detailed data of the earthquake. However, they will only need the necessary data from earthquakes happening close to them and only require intensity data of the point in interest (e.g. the power plant). To put this into content, an earthquake happening 1000km away from them does not need to be fed into their system, while they would like to see the intensity of the shock and the arrival time of the seismic waves to decide the actions. In fact, the author really likes investigating on the rail industry, whose infrastructure could be greatly affected by earthquakes.

Table 1 compares relative features of these two target users/clients.

Feature	Primary Target	Secondary Target
Description	Passionate Geologists	Earthquake-Sensitive Industries
Age	Varies (Middle School – Adults)	Work Age
Reason	Monitor Live Latest Earthquakes	Monitor Risks to Infrastructure
Proficiency	Varies (Beginner – Amateur)	Trained Professional
General Requirements	Monitor Overall Movement	Alert about intensity and arrival time at specific points

Table 1: Comparison of target users and clients

1.4 Research Methodology

1.4.1 Client Interview

The author interviewed my friend Wesley Ma, who is a passionate geologist on earthquake studies and also monitor earthquakes regularly.

1. *Which earthquake monitoring apps do you use?*

Response: JQuake, SREV, Quarog, KEVI, Kyoshin-Monitor (Support discontinued), Kiwi Monitor (Support discontinued)

2. *Do you subscribe/pay to services such as DM-D.S.S. to use earthquake monitoring apps, and do you think it is worth the price?*

Response: Yes. However DM-D.S.S. is a little bit expensive. However, the price becomes more affordable considering the information provided by the subscription.

3. *Do you watch YouTube livestreams on earthquake monitoring?*

Response: I do not usually watch the live streams, as almost no one who has already has a monitoring app will use the stream. They provide mostly the same information as the applications, just real-time streaming the windows.

4. *Why do you use earthquake monitoring apps?*

Response: To monitor the earthquake. This is derived from my interest in broadcasting culture in Japan. This eventually led me to be intrigued with the development of earthquake monitoring technologies and theories in Japan.

5. *How often do you use earthquake monitoring apps (e.g. all the time/after school/only after big earthquakes)?*

Response: After big earthquakes. But I usually open one or two apps for all-day monitoring to catch potential major (or medium) earthquakes.

6. *Describe the advantages and disadvantages of each of them, mentioning the specific features.*

Response:

- SREV is a good one, but it only available in a browser with no app.
- Kyoshin-Monitor and Kiwi Monitor are relatively more stable, but the source is not the same as that from the former two apps, and its support is also discontinued.
- Quarog has weak response time and interacting interface.
- JQuake is the most developed app, which includes nearly all functions that can be thought of. However sometimes the connection of WebSocket is unstable.
- KEW does not have the sound files configured by default. Some information are not displayed clearly enough.

7. *What features do you use the most/least?*

Response: Basic earthquake notifications, and should be including sufficient and prompt information.

8. *What features are redundant in the earthquake monitoring apps you use?*

Response: KEVI has a weather monitoring function, which could be redundant. But that could be caused by the different purposes of the app. Hence, no further comments. It is not a bad thing.

9. *What are the critical features of an earthquake monitoring app?*

Response: To provide accurate, prompt and detailed information according to the source released by the JMA, the information display interface should be easy to read and understand. This is not only helping the people who like to monitor, but more importantly,

provides the easiest way to the people who really need to seek information to minimise the harm brought by earthquakes and successive disaster.

10. *What additional features would you like to have in those existing apps?*

Response: Summarise all the useful features from different apps into one app. Stable connection. Nothing else.

1.4.2 Existing Applications and Solutions

Based on the applications the author uses and the feedback from interviewee, there are the following commonly-used applications:

- JQuake <https://jquake.net>
- Scratch Realtime Earthquake Viewer (SREV)
<https://kotoho7.github.io/scratch-realtime-earthquake-viewer-page/>
- Kyoshin EEW Viewer for Ingen (KEVI) <https://svs.ingen084.net/kyoshineewviewer/>
- Quarog <https://fuku1213.github.io/quarog-site/>

Supported platforms of those apps are listed in Table 2. In particular, note that SREV is a web-based and GitHub Pages-hosted application therefore supporting all platforms. KEVI is written in .NET Framework and supports the second most platforms, with JQuake not supporting Linux and Quarog only supporting Windows.

Platform	JQuake	SREV	KEVI	Quarog
Windows	✓	✓	✓	✓
macOS	✓	✓	✓	
Linux	✓	✓	✓	
Android/iOS		✓		

Table 2: Supported platforms of existing solutions

An overview feature table of monitoring is analysed in Table 3. In particular, note that due to the nature of SREV being Scratch-programmed and web-based, it reached a special agreement with DM-D.S.S. to use the API without the need of all users paying for this, since it is hard to integrate such function into a web application.

Quarog is a relatively new application and only supports basic functionalities of EEW Viewing and past earthquake listing, as shown in Figure 2. JQuake is solely dedicated to earthquake and tsunami monitoring as shown in Figure 1, while KEVI has features like rain clouds map and natural disaster warning which is beyond the scope of this analysis (which is a burden to users only requiring earthquake monitoring and a waste of storage space).

It is worth noting that for SREV, DM-D.S.S. is integrated with special permission with no login required. For JQuake, although it has tsunami warnings/special warnings, the tsunami forecast is not available.

There are also a variety of configuration options available for all apps, as listed in Table 4. Both KEVI and Quarog supports the adjustment of the colour theme, and Quarog even supports changing the style of how blocks are displayed and coloured as shown in Figure 3 and 4. Playing a sound on the speaker is also common among the apps to remind the user of earthquakes.

Feature	JQuake	SREV	KEVI	Quarog
DM-D.S.S. WebSocket Support	✓	✓	✓	✓
Real-time Sensor Data	✓	✓	✓	
Vibration Alert	✓	✓	✓	
Past Earthquake List	✓	✓	✓	✓
Past Earthquake Details		✓	✓	✓
Tsunami Warning	✓	✓	✓	
Realtime EEW	✓	✓	✓	✓
Calculated Seismic Wavefronts	✓	✓	✓	✓
User-Defined Key Monitor Point	✓		✓	
Sub-Map for the Okinawa Area	✓		✓	
Replay	✓		✓	

Table 3: Feature comparison in monitoring of existing solutions

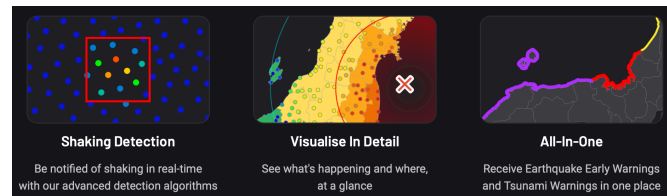


Figure 1: Feature introduction of JQuake, screenshot from website.

Feature	JQuake	SREV	KEVI	Quarog
DM-D.S.S. Login	✓		✓	✓
Sound Alert	✓	✓	✓	✓
System Notification			✓	
Colour Theme			✓	✓
Map Colouring Style		✓		✓

Table 4: Feature comparison in configuration and customisability of existing solutions

1.5 Features of proposed solution

Based on the potential user/client interview results, and the research into the existing solutions, the following key features should appear in the solution, since they are essential to earthquake monitoring applications:

1. DM-D.S.S. Login functionality (for the data source)
2. Real-Time Sensor Shake Intensity Data (w/ vibration alert)
3. EEW Visualisation (w/ Calculated Seismic Wavefronts)
4. Past Earthquake List (w/ Option to review details)
5. Tsunami Warning Visualisation (w/ Related Sounds)

However, due to the limitation of time and the difficulty of implementation, the following functionalities are not the key to implementation, which include mostly the customisation parts, ranked in decreasing importance:



Figure 2: Feature introduction of Quarog, screenshot from website. Top-left: Past earthquake information; Top-right: Real-time EEW; Bottom-Left: Past earthquake list; Bottom-Right: Details of EEW.

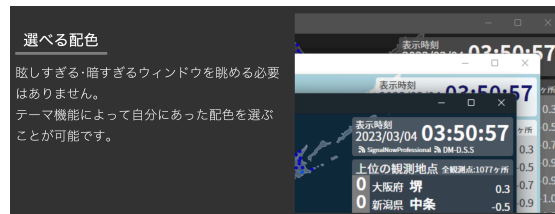


Figure 3: Customisable colour scheme of KEVI, screenshot from website.

1. User-Defined Key Monitor Point
2. Customisable Sound Alert
3. Customisable Colour Theme
4. Customisable Map Colouring Style

The following features might not be available due to the time constraints and the complexity behind such system:

1. Replay (due to a server needing to store past data)



Figure 4: Customisability of Quarog, screenshot from website.

2. Sub-Map for the Okinawa Area (due to the difficulty in implementatino)
3. Map Zooming Feature (due to the complexity in the map colouring functionality)

With those key features implemented, the application should mirror all essential functionalities of a earthquake monitoring application, as compared to the four exsisting solutions above.

1.6 Critical Path

The functionality of this application can essentially be divied into $1 + 2 + 1 + n$ steps, where the 1 is to receive information from the API/WebSocket provided by DM-D.S.S., and 2 is to, briefly saying:

1. Real-time Monitoring: Produce a real-time map to monitor real time vibration and plot real-time EEW/tsunami warnings
2. Past-earthquake Viewing: Produce a menu to select an earthquake to display information on the map.

The next 1 is to merge these two functions, specifically the functionality to switch back to the real-time monitoring option immediately when a new EEW is released (to make sure the user does not miss any information on realtime EEWs while looking at past earthquake information).

The final n is to implement a setting page for customisation, and some extra additional features.

Part 1. DM-D.S.S. WebSocket Connection.

1. Investigate into the list of APIs/WebSockets that should be used by the application.
2. Implement classes/DTOs to convert the information into C# objects.
3. Implement classes to transfer real-time WebSocket XML/JSON to the classes and DTOs.
4. Use simple API Key to test the functionality of this sub-system.
5. Implement OAuth 2 login to reduce complexity and increase security.

Part 2a. Real-time Monitoring Map.

1. Implement a Japan map in the application.
2. Achieve and store the locations of the monitoring points in Japan, using necessary information from JMA and DM-D.S.S.
3. Colour the monitoring points using a colour scheme on the map.
4. Investigate into and implement an algorithm to detect shake in a certain area of the map.
5. Achieve and store the names of areas of earthquake epicenters from JMA.
6. Investigate into and implement an algorithm to calculate seismic wave fronts.
7. Plot and display real time EEWs, considering special cases such as:
 - Cancellation of EEW;
 - Upgrade from EEW Forecast to EEW Warning;
 - Multiple Earthquakes (hence displaying epicenters with labels).
8. Implement functionality of colouring areas on the map with the maximum expected intensity.
9. Achieve and store the names of shorelines from JMA.
10. Plot and display real time tsunami warnings.

Part 2b. Past-earthquake Viewing.

1. Implement a side-list of a list of past earthquakes.
2. Provide a button to view the detailed information on the earthquake (e.g. magnitude, epicenter, time).
3. Implement functionality to plot the detected maximum intensities on the map, from where they are detected.
4. Provide external link to view earthquake in the JMA website.

Part 3. Joint functionality.

1. Provide sidebar to switch between real-time monitoring and past-earthquake viewing.
2. Implement automatic functionality to switch back to real-time earthquake monitoring when a shake over a certain magnitude is detected, or an EEW/Tsunami Warning is being published.

Part 4. Setting page for customisation and some more.

1. Implement customisable voice and sound playing (e.g. when EEW (over certain magnitude) is published, when an earthquake information detail is received, etc.)
2. Implement customisable colour scheme for the colouring of different intensity scales, with several built in default.
3. Implement key monitoring point with special warnings when an earthquake with more than a certain intensity is expected to hit that point.
4. Implement a map-zooming feature.
5. Implement a sub-map for the Okinawa Area.

1.7 Requirements Specification

A detailed specification is defined here that is to be aimed to be fulfilled at the end of the project, split into four sections:

1. **DM-D.S.S. Functionality** – Corresponding to **Part 1**, in Table 6
2. **Real-Time Earthquake Monitoring** – Corresponding to **Part 2a**, in Table 7
3. **Past-earthquake Viewing** – Corresponding to **Part 2b**, in Table 8
4. **GUI Design** – Corresponding to **Part 3, 4**, in Table 9

The objectives here are SMART, meaning they are specific, measurable, achievable, realistic and timely.

Note that those marked with an * means they are optional.

Table 5 for types of testing. [M] afterwards will stand for a necessary manual testing (i.e. specific user inputs), while (M) will stand for supplementary manual testing (i.e. will have stand-alone tests as well as manual tests).

Test Method	Abbr.
Unit Testing	UT
Integrated Testing	IT
Functional Testing	FT
Performance Testing	PT

Table 5: Measurement methods

Req. №	Description	Success Criteria	Testing
1(i)	Login to DM-D.S.S. using an API Key	Login successful	UT [M]
1(ii)	Call HTTP-Based APIs	Successful calls	UT (M)
1(iii)	Connect to WebSocket Data Feed	Successful calls	UT [M]
1(iv)	Obtain stable connection on WS	Connected 30min	PT (M)
1(v)	Parse JSON and XML into C# objects	Information successfully parsed	UT
1(vi)	Exception handling for incorrect JSON and XML	Exceptions thrown	UT
1(vii)	Exception handling for failed connections	Exceptions thrown	UT (M)
1(viii)	Integrated functionality from login to parsing	Correct objects created	IT, FT [M]
1(ix)*	Login to DM-D.S.S. using OAuth2	Login successful	UT (M)

Table 6: Requirements for Part 1

Req. №	Description	Success Criteria	Testing
2a(i)	Login to DM-D.S.S. using an API Key	Login successful	UT [M]
2a(ii)	Call HTTP-Based APIs	Successful calls	UT (M)
2a(iii)	Connect to WebSocket Data Feed	Successful calls	UT [M]
2a(iv)	Obtain stable connection on WS	Connected 30min	PT (M)
2a(v)	Parse JSON and XML into C# objects	Information successfully parsed	UT
2a(vi)	Exception handling for incorrect JSON and XML	Exceptions thrown	UT
2a(vii)	Exception handling for failed connections	Exceptions thrown	UT (M)
2a(viii)	Integrated functionality from login to parsing	Correct objects created	IT, FT [M]
2a(ix)*	Login to DM-D.S.S. using OAuth2	Login successful	UT (M)

Table 7: Requirements for Part 2(a)

Req. №	Description	Success Criteria	Testing
2b(i)	Login to DM-D.S.S. using an API Key	Login successful	UT [M]
2b(ii)	Call HTTP-Based APIs	Successful calls	UT (M)
2b(iii)	Connect to WebSocket Data Feed	Successful calls	UT [M]
2b(iv)	Obtain stable connection on WS	Connected 30min	PT (M)
2b(v)	Parse JSON and XML into C# objects	Information successfully parsed	UT
2b(vi)	Exception handling for incorrect JSON and XML	Exceptions thrown	UT
2b(vii)	Exception handling for failed connections	Exceptions thrown	UT (M)
2b(viii)	Integrated functionality from login to parsing	Correct objects created	IT, FT [M]
2b(ix)*	Login to DM-D.S.S. using OAuth2	Login successful	UT (M)

Table 8: Requirements for Part 2(b)

Req. №	Description	Success Criteria	Testing
3(i)	Login to DM-D.S.S. using an API Key	Login successful	UT [M]
3(ii)	Call HTTP-Based APIs	Successful calls	UT (M)
3(iii)	Connect to WebSocket Data Feed	Successful calls	UT [M]
3(iv)	Obtain stable connection on WS	Connected 30min	PT (M)
3(v)	Parse JSON and XML into C# objects	Information successfully parsed	UT
3(vi)	Exception handling for incorrect JSON and XML	Exceptions thrown	UT
3(vii)	Exception handling for failed connections	Exceptions thrown	UT (M)
3(viii)	Integrated functionality from login to parsing	Correct objects created	IT, FT [M]
3(ix)*	Login to DM-D.S.S. using OAuth2	Login successful	UT (M)

Table 9: Requirements for Part 3

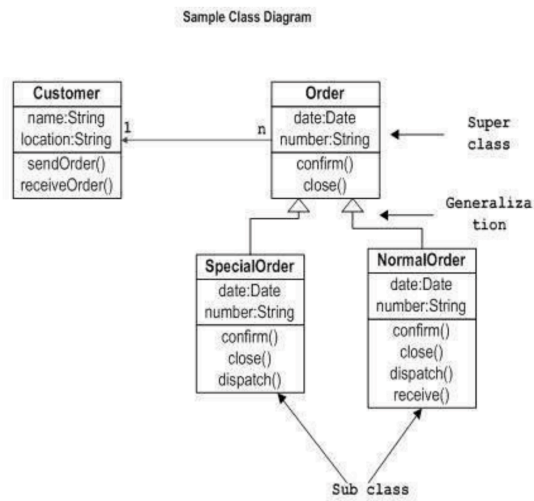


Figure 6: Class Diagram.

2.4 Hardware Software Requirements

Draw up a hardware and software specification for items that are required.

3 Technical Implementation

3.1 Key Code Segments

3.1.1 Data structures

Implementation of ADTs and OOP Classes to be demonstrated.

3.1.2 Modularity

Code should be created and tested in separate modules that are integrated later. Use sub-headings for each module, define the purpose of the module, and show unit testing of the module.

3.1.3 Defensive Programming/Robustness

Exception handling

4 Testing

Consider how you will test your project. You should devise a test strategy that encompasses a range of methods.

4.1 Test Strategy

- Unit testing (of individual functions)
- Integration testing (e.g. different modules/class files)
- Robustness (demonstrating defensive programming skills/exception handling)
- Requirements testing (against your initial requirements - a table with test number, description, test data, expected result, evidence (screenshot/video time link) would be suitable)
- Independent end user beta testing (this will assist with your evaluation)

4.2 Testing Video

- You can include a video to assist (but you will need to reference the timepoint at which relevant evidence appears)
- If you include a video you will need to have it publicly available.
- It is suggested that you include a QR code in your testing to give a link to it the video (for the moderator) rather than just giving a long URL on its own.

4.3 System Tests (against original requirements specification)

You need to give evidence in support of requirements that have been met e.g. reference to a relevant test/screenshot/relevant code.

Requirement №	Description	Success Criteria	Tests + Evidence

Table 10: Table of Tests.

5 Evaluation

5.1 Requirements Specification Evaluation

Personal evaluation

- Copy and paste your original requirements from your project analysis
- You need to review each requirement and comment objectively on whether it was *fully met/partially met/not met*.

Requirement №	Description	Success Criteria	Fully/Partial/Not met (Reflective Comment)

Table 11: Table of Evaluation.

5.2 Independent End-User Feedback

End user/client evaluation

- there **must** be meaningful end user feedback
- You should hold a review meeting with your end user
- Write down any key feedback that they give you. E.g. Agreement that a particular requirement has been met/comments as to aspects that they find sub-optimal/comments as to additions they would like to see

Requirement №	Description	Acceptance Y/N	Additional Comments

Table 12: Table of Feedback.

5.3 Improvements

You need to give consideration to a number of potential future improvements that could be made. They may arise from either your experience or from feedback given to you by your end user. Ideally at least one should be in response to end user feedback.

- Write a paragraph for each potential improvement/change
- The improvements/changes could result from additional functionality that has been identified as being beneficial or could be as a result of required efficiencies if some processes are clunky or require faster run-times

- You should then comment on how the proposed change could be implemented moving forward. i.e. what would need to be changed/developed and how? You are not expected to actually make any changes; just comment on the possibilities.

6 Code Listing

List of Tables

1	Comparison of target users and clients	5
2	Supported platforms of exsisting solutions	7
3	Feature comparison in monitoring of existing solutions	8
4	Feature comparison in configuration and customisability of existing solutions	8
5	Measurement methods	12
6	Requirements for Part 1	12
7	Requirements for Part 2(a)	13
8	Requirements for Part 2(b)	13
9	Requirements for Part 3	14
10	Table of Tests.	18
11	Table of Evaluation.	19
12	Table of Feedback.	19

List of Figures

1	Feature introduction of JQuake	8
2	Feature introduction of Quarog	9
3	Customisable colour scheme of KEVI	9
4	Customisablity of Quarog	10
5	Hierarchy Chart.	15
6	Class Diagram.	16