University of Cambridge

Computer Science Tripos
Part 1B Group Project 2013-2014
Team Foxtrot - Money World

Progress Report

Daniel Low
Darren Foong
Jovan Powar
Samuel Haines

Last Revised: February 13, 2014

1 Summary

Development of the Money World app is going all right except for some minor set backs.

Firstly, one of our group members has had to leave the group due to personal reasons. As a result, members' tasks had to be reallocated.

Secondly, The first prototype was delayed because we experienced difficulties setting up the development environment. Some of us were using iPhones and some who were using Android phones were also having issues launching the application on their phones.

At the time of writing, a prototype has been completed, and will be tested before being demonstrated live at the client meeting on Friday 14 Feb. This prototype largely follows the interface described in the specification, with some visualisation models that will be described below.

Subsequent iterations will see more visualisation models, mainly a chart for historical view and a comparison page, and some UI/UX refinements.

Two weeks since the inception of the start of the project, our group has:

- Implemented most of the backend for data retrieval.
- Implemented a basic user interface to navigate the app.
- Developed some visualisation models for use.

These will be elaborated in the following sections.

2 Components

2.1 Data retrieval component

This server-side component has been implemented in Java using the Jersey RESTful Web Services framework with a MySQL database. It is currently running on Google App Engine at `http://money-world.appspot.com` with a simple API to obtain data in JSON format.

The component executes SQL queries on the MySQL database to obtain Java objects, which are further marshalled into JSON output according to a schema.

The aggregator has not been implemented; currently the database contains only data from the World Bank. Event data for countries will be added to enhance visualisations.

The database schema is as follows:

- **Country** CountryCode, CountryName, Region, IncomeGroup, CountryNotes
- **DataSet** DataSetCode, DataSetName, DataSetNotes, Source
- **DataPoint** DataSetCode, CountryCode, Year, Value,

- **Visualisation** VisualisationID, Type, ExtraData
- **VisualisationCountry** VisualisationID, CountryCode
- **VisualisationDataSet** VisualisationID, DataSetCode
- **Comment** CommentID, CommentBody, CommentUser, VisualisationID, Created, ExtraData
- **CountryCode** CountryCode2, CountryCode3

2.2 Data explanation component

Not implemented yet (as per project plan).

2.3 Data visualisation component

This component's function is to take a set of data and output it in one of the following visualisations.

2.3.1 Map view

This view will display a world map of the region the country is in (users will have the ability to switch regions if they wish to). For our prototype, the regions are western and southern Africa. Countries in this region will have a colour associated with it which correlates with the economic data currently being visualised.

Navigation around the map is done by panning and using two finger pinching for zooming. This allows the user to select even the smallest country.

The map includes a legends which will appear when a button is pressed, this will give information about what range of values each colour meant.

A population count is displayed at the bottom right. This will display the population of the country when the user hovers (on the computer) or taps (on the phone) on a country. It is represented in a readable form such as "5 million" instead of 5000000.

Maps of different region are separated so that only a particular region is visible on the screen. This is to allow localised comparison.

A slider is added at the bottom which is used to adjust the year. The map will be redrawn with new data based on the year.

Testing has been done by visually checking due to the nature of the code produced. Automated unit testing has not been implemented for this as it is small enough to be tested manually.

The following has been visually verified

- Values of the indicator corresponds correctly with the colour of the region
- Values are updated when the slider change

- Legends are updated when there are changes
- Population count is transcribed correctly

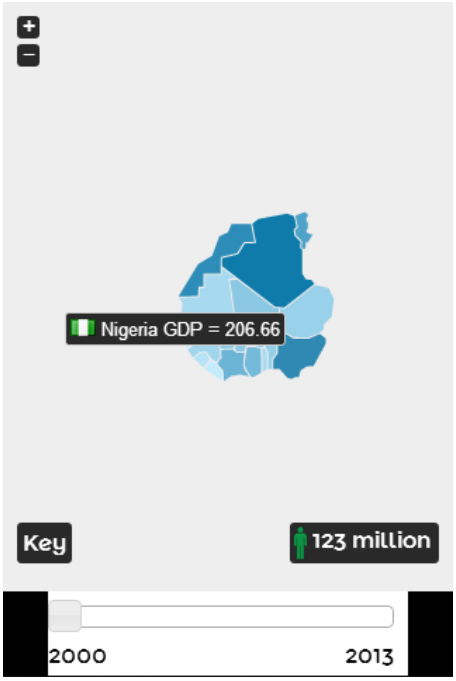

Figure 1: Map view

2.3.2 Stack View

This is a view where indicator values are represented by the size of the rectangles. The country with the smallest value will be at the front of the page, and it will be stacked on top of countries with larger values.

The input is a JSON file with a similar format to the one used by the map view.

The output is automatically generated rectangles identifiable by country flags.

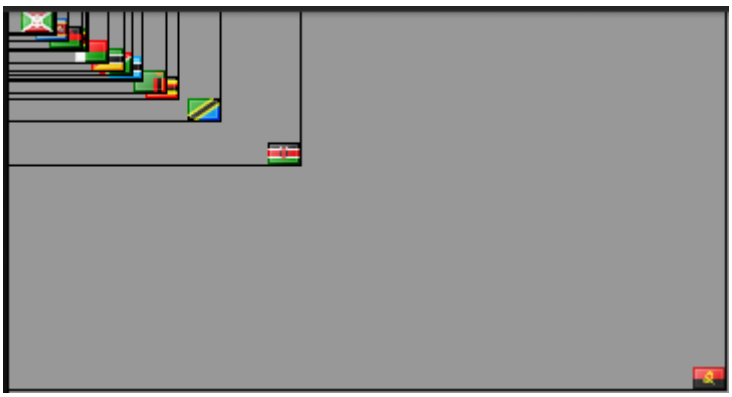

Figure 2: Stack view

2.3.3 Grid view

After further consideration, this view may not be used at all because it isn't entirely intuitive how the economic data are visualised other than being outputted as text.

This is a view where country's economic data is represented in some manner within individual cells that are automatically generated based on the input given.

The input is a JSON file with a similar format to the one used by the map view.

2.3.4 Bubble chart view

This view's development is delayed.

2.3.5 Line chart view

This view's development is delayed.

2.4 Comment display and input component

This component is dropped because we have decided it was unnecessary during the first client meeting. This is combined with the sharing component.

2.5 Comment storage component

This component is dropped because we have decided it was unnecessary during the first client meeting. This is combined with the sharing component.

2.6 Sharing component

Not implemented yet (as per project plan).

2.7 UI and UX component

The frontend runs on the Sencha Touch framework and Cordova to run on Android phones. A basic user interface has been implemented that can easily incorporate visualisations. Basic user details can be stored and retrieved for the application's purposes (country, region).

Further improvements required are styling and theming of the interface, with extra graphics and icons to enhance the apperance.

The frontend is build according to the Model-View-Controller (MVC framework). As of writing, the components are:

- **Models** Setting, Country

- **Views** Init, Main, Settings, Overview, DetailedView, ComparisonView, MainTileView, MapView
- **Controllers** Init, Main, Settings, Overview, MainTileView, MapView
- **Stores** Settings, Countries