Architectural Requirements, Architecture Design and Non-Functional Requirements

Project: Insurance profiling from social media
Client: RetroRabbit

Team: Valknut Solutions

Version: 1.3

- 13054903 - Charl Jansen van Vuuren
- 13044924 - Kevin Heritage
- 13176545 - Quinton Weenink

Department of Computer Science

October 20, 2016

Contents

Revision History

Revision	Date	Author(s)	Description
1.0	27.05.2016	CJvV,KH,QW	Created
1.1	27.07.2016	CJvV,KH,QW	Grammar errors, Introduction, Scope, Vision, Access channel, Quality requirements, Integration Requirements (API, Protocols, API Quality requirements), Architectural tactics(Security) and Architectural pattern changes
1.2	07.09.2016	CJvV,KH,QW	Updated introduction, scope, Access channel requirements, Integration quality requirements, architectural Tactics, Architectural pattern
1.3	05.10.2016	CJvV,KH,QW	Updated front-page with logo, updated introduction, Vision, Scope, Access channel requirements, Quality requirements, Tactics, Overall flow of the document

Introduction

Insurance profiling and risk analysis often require large amounts of data to generate in-depth profiles, our project aims to eliminate the need for vast amounts of data gathering by utilising a user's social media information, based on advertisement forms and the integration of a Facebook and Wechat messenger bot.
This includes the managing and processing of these leads from the marketing and analysis viewpoints.

Scope of system

The project as it is currently specified is an application that will allow insurance companies the ability to get data from lead data provided via lead advertisements on Facebook. A user fills in a Facebook advertisement to purchase short term insurance based on Facebook's advertisement audience algorithm.
Further access points are available by means of a personalized Facebook messenger bot and Wechat messenger integration.
The use of messenger bots allow for a more personalized look and feel, as the user can supply information by means of talking to the bot.

This data will be used to charge the users for the insurance that they have signed up for while also providing the application with some basic data that can later be used to analyse the users and the data they provided. An analyst would be able to log into our system and generate risk statistics for a certain advertisement based on certain criteria, this information gives the insurance company the ability to charge and adapt their sale and costs based on popular advertisements.

This information will be available based on API calls to the database or through a Web front-end for analysts.
A live dashboard for marketing follow ups will allow marketers the ability to see unprocessed leads and live graphs.

Vision

We seek to implement a real time marketing and lead gathering application with multiple integration points with social media and other formats.
A business can manage and respond to leads and analyze their current customers and audience.
We expose and link companies to their future customers in an easy and pluggable manner.

Architectural Requirements

Access channel requirements

User:

1. A user will fill in an advertisement form on Facebook, or chat with a messenger bot to supply the information specified and submit this data.
2. This data can also be filled in via the website form.
3. The necessary data will be gathered and saved in the database.
4. The system will analyze the data and send the user a response via email if they qualified for the insurance.
5. A sales person will contact the specified user with further instructions.

Analyst:

1. An analyst would log into the system with a required username and password authorization
2. Selection criteria to generate a report will be presented to the analyst.
3. The analyst will be able to filter the report/graph/statistics based on certain specified criteria.
4. Options for exporting the data will be available in different formats.

Marketer:

1. A marketer will supply the necessary authorization to access the dashboard.
2. A number of live graphs will be presented to the marketer to see new leads and user signups.
3. The marketer will be able to see a list of unprocessed leads to follow up on

Further access channel specification include:

- The website will be accessible from, and optimized for all web-browsers including mobile phone browsers.
- The possibility of developing a mobile application will be considered as per the client's request.
- The website will be accessible via the majority of operating systems if such an operating system has access to a supported web-browser.
- Facebook advertisement integration will be accessible via the normal Facebook access channels.
- Facebook messenger integration will be accessible via the Messenger application.
- Wechat integration will be available via the Wechat messenger platform.

Quality Requirements

The tactics used to address these quality requirement issues are discussed in Section 5.1

Performance

- A user should be able to request a quote in less than 5 minutes. This is easily achieved by means of the Facebook advertisement and will only be user-network dependent.
- Once the REST API call is made, it saves the user's data in our database, increasing the efficiency of future requests and processing of the data.
- The need to do external API calls will be minimized, increasing performance.
- Tactics: Section 5.1.2

Security

- Security is our most important architectural requirement. A user's personal information is used to generate these risk profiles and as a result the user will trust that this information is not shared with other parties, and properly secured.
- Only authorised persons will have access to generated statistics and access to the database and API will be restricted to authorized analysts and administrators only.
- The Heroku server's adds an extra layer of security with regards to random generation of usernames and passwords for access to the database.
- Tactics : Section 5.1.5

Scalability

- Since this project is a web-based solution, the possibility of multiple concurrent users should be considered.
- The server should account for a vast amount of concurrent users.
- As per integratability 4.2.4, the ability to change the risk analysis algorithm as needed should be considered for a future scalable solution.
- Tactics : Section 5.1.7

Integratability

- The project will integrate with Facebook and utilise it as the primary data provider.
- The ability to integrate with other social media platforms should be considered and modularised accordingly, to ensure seamless future integration.
- The ability to change the risk analysis algorithm as needed should be considered.
- Integration includes the connection from the website to the profiling engine and back to the website as a report.
- Integration with Facebook's advertisements API, Wechat messing API and Facebook's messenger API form major parts of this release.
- Tactics : Section 5.1.3

Reliability

- Since our solution is mainly web-based, the platform as a service (PaaS) offered by Heroku will ensure the website is always up to date and reliable.
- Tactics : Section 5.1.6

Maintainability

The system will make use of a database with massive amounts of data.
To ensure optimal performance, this data will need to be maintained and normalised on a regular basis.

Auditability

- All actions performed in the system should be traceable to the user that performed them.
- The user's IP will be logged as to have a form of accountability in the persistence of the data.
- Tactics: Section 5.1.4

Cost

The majority of our platform is open-source, except for:

- Heroku hosting if the commercial version is used
- Travis CI if the commercial version is used
- Facebook's advertisement campaigns are billed monthly if the advertisement is to be live.

The current free versions of these platforms are used.
The Facebook advertisements are currently not live as this project is still in the development stage.

Usability

- The platform is being developed with efficiency in mind, as a result the input and response of the website should be visually pleasing and simple to use.
- Further integration with the Facebook ad system will increase usability, as the user will be able to generate a report from their Facebook dashboard, without the need to log in to our website.
- The Facebook messenger integration allows a more fulfilling user experience as it eliminates the normal form structure of capturing data.
- Tactics : Section 5.1.8

Flexibility

- The system must be usable on any internet browser (including mobile phone browsers).
- The system must include the ability to change the risk analysis algorithm. Insurance profilers can use this to their advantage to generate more thorough risk profiles.
- Tactics : Section 5.1.1

Compatibility

- The use of Travis CI tests ensures the latest features are compatible with the specified versions of NodeJS.
- Unit tests constantly test the system's compatibility as part of the continuous integration model.

Integration requirements

API specifications

- The Facebook advertisement and marketing API form a major part of this project.
- The Facebook Pages API forms part of this.
- The use of Facebook's Graph API by means of the Node package.
- An alternative is to use Curl based calls to interact with these APIs.
- The method of interacting with the API will be to parse the JSON string that will be returned by the Facebook server.
- It is also possible to get the access token using this method (a string of characters giving you access to a persons account with the permissions specified).
- Facebook messenger bot integration is supplied by means of the Facebook messenger api Facebook Messenger
- Wechat integration is supplie by means of the Official account developer platform API, for Wechat. Wechat API

Regarding the use of access tokens, a Facebook page Access token and Facebook user access token will be used. These Access Token should also have the longest lifetime possible to be able to update user data for as long as possible.

Access Tokens

- The access tokens generated via web logins are short-lived, but one can convert them to long-lived tokens.
- Apps with Standard access to Facebook's Marketing API when using long-lived tokens will receive long-lived tokens that don't have an expiry time.
- For security to validate some data we may need to gain Application access tokens. These tokens are only for the developers of the application not for general public.
- To generate an app access token:

```
/oauth/access_token?
     client_id={app-id}
    &amp;client_secret={app-secret}
    &amp;grant_type=client_credentials
```

- Once again the app secret should be kept safe and hidden from users

Protocols

The system will include the use of these protocols:

- HTTP/HTTPS - (Secure) Hypertext Transfer Protocol
- TCP/IP - Transmission Control Protocol/Internet Protocol
- FTP(Possibly) - File Transfer Protocol
- SSL - Secure Socket Layer
- SMTP - Simple Mail Transfer Protocol(with SSL)

The request for comment pages of these protocols can be accessed via:

- HTTP
- HTTPS
- TCP
- IP
- FTP
- SSL
- SMTP

Integration Quality Requirements

The below mentioned use of the word Facebook is related to the Facebook advertisement integration and the messenger platform

As mentioned in 4.2.4, integration with Facebook will be a major feature, as such this integration is subject to quality requirements in the form of:

- Reliability - The Facebook API, Wechat API and Messenger API server must be available at all times as this forms a major part of the system.
- Auditability - The use of the Facebook developer application ID allows for auditability as to the use of the Facebook API in our system. The developer application ID allows Facebook to obtain information on the current use of their API with regards to our system. Wechat uses their own form of auditability similar to the Facebook method.
- Performance - Our system's performance in certain areas, is limited to the use of the above mentioned APIs. The API systesm must respond before our system can analyse the data.
- Security - Facebook's API has security measures in place to ensure that the Advertisement's data isn't accessed without authorization. This consent is granted by means of a permission request to the Analyst's Facebook page before we can obtain any information. The safeguard of this information is a major concern as mentioned in 4.2.2.
- Maintainability - The system must be maintained on a bi-yearly basis as to ensure the latest version of the API is used and no deprecated functions are used. The use of deprecated functions can cause problems in future use of the system.

Architecture Constraints

Our client limited our technologies to:

- An ASP.net web solution with a Microsoft SQL Server database system.
- A NodeJS web solution with a PostgreSQL database system.
- The use of a platform as a services (PaaS) hosting solution, Heroku.
- We have the freedom to use the Facebook SDK in any language except for PHP as mentioned below.

Our client specifically constrained the use of:

- PHP code in any way.
- MySQL, NoSQL database systems.

Other constraints in terms of architecture include:

- Browser independence, to ensure any web-client can make use thereof.
- Operating system independence.
- The system should be as time efficient as possible
- Mobile web-browser compatibility

Initial Architecture Design

The below mentioned architecture specifies an initial architecture design from a high-leve l standpoint, of the system as it is currently understood at this stage in development and design. Any and all architectural components, tactics and responsibilities listed are subject to change and are at a level which attempts to provide the best description of what is currently intended to be the insurance profiling system.

Architectural tactics

Due to the fact that the underlying architecture has of yet not been clearly specified, the investigation into the best suited tactics for this architecture can not be concretely concluded. The following is by no means a representation of the implemented tactics but it is currently what best helps the system achieve its quality requirements. The below listed tactics are currently the main tactics we aim to implement or the responsibility of the dependencies we aim to utilize.

Modifiability and Flexibility

What is certain, is that the above mentioned quality requirements will need to be addressed in a manner which allows for a system that is both highly modular and configurable, in order to allow the analysts of the data set to properly assess and tweak the tools that we aim to provide to them.

- Flexibility support: automated builds, automated testing and automated environment configuration
- Service provider flexibility: dependency injection
- Process flexibility: pipes and filters, responsibility localization

Performance

- Object Relational mapping for database calls increase performance
- NodeJS makes use of Asynchronous callbacks to increase performance and not slow down other calls

Accessibility and Integratability

- Providing access: providing proxies, supporting standard communication protocols (REST calls)

Auditability

- Auditability closes relates to Security 5.1.5
- Authentication and authorization tactics in general.

Security

- Resisting attacks: authentication, authorization, minimizing access channels, minimize complexity and enforcing secure defaults
- Detecting attacks: auditing, logging and integrity checking
- Recovering: restore states, dropping
- Hashing of passwords : Password hashes are created based on the SHA512 algorithm with a length 16, random generated salt.
- Adding secret token to access tokens : Front-end tokens are generated based on a secret key word and SHA256 encoding.

Reliability

- Preventing faults: resource locking, testing framework, contracts based
- Detect faults: logging, deadlock detection and error/exception communication
- Recovery: rollback, backups and passive redundancy
- Version based branching to enable rollback functionality

Scalability

- Resource demand: indexing, paging, query optimization

Usability

- The use of virtualization increases the overall reliability and usability of the system. Most hosting platforms makes use of virtualization platforms.

Architectural components addressing architectural responsibilities

- Providing human access channels and external systems web access to the system services which in turn grants access to a web services framework such as a custom REST api,
- Providing users access through a browser to a web application framework. Node.js has such a framework called express.js and AngularJS.
- In order to provide hosting for business logic processes, an application server will be hosted by Heroku.
- Persistence of objects will be stored in an object-relational database management system called PostgreSQL.
- The responsibility of providing the persistence provider access to the persistence API by means of the Sequelize ORM.

Infrastructure

Architectural (structural) patterns

- In order to promote flexibility as well as testability, a hybrid pattern will be used, with key-traits from the layered architectural pattern and the Model-View-Controller pattern.
- The layered approach allows for future development to change tactics (business logic tactics) and allows for the decoupling of different functions.
- The Model-View-Controller pattern allows for a understandable and testable system, with proper modularization. MVC includes the ability to generate new views for the web based front-end efficiently. See Figure 1
- Decoupling is crucial in order for the system to remain within its architectural concern boundaries, which will rely heavily on a modular configurable system.

Concepts and constraints for application components

Application components will not be deployed at the first level of granularity. Any application components will be hosted within the architectural components.

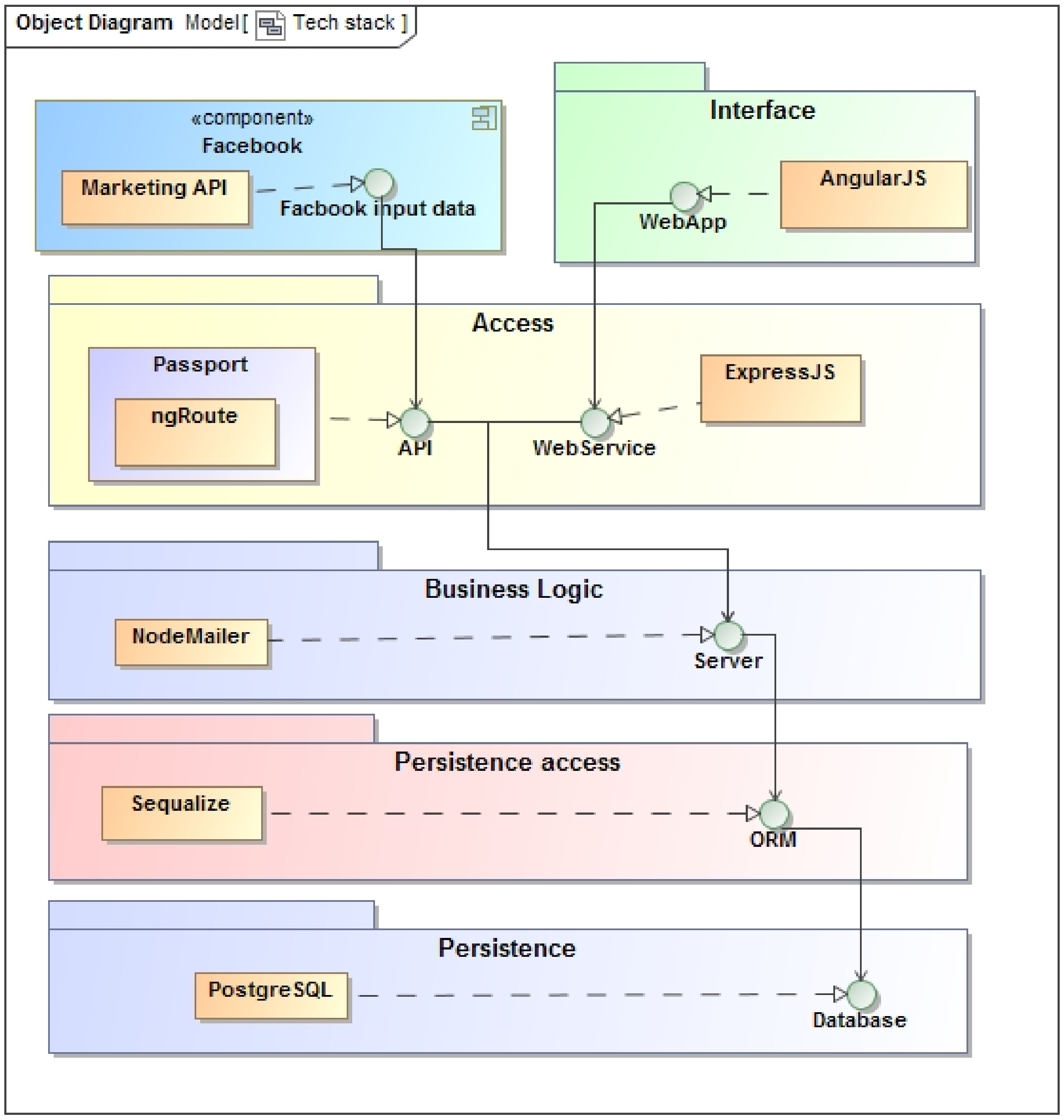

Figure 1: Hybrid architectural pattern : Insurance Profiling