

# Bliss Recruitment App Requirements Specification

## Change Log

Date	Author	Changes
19 September 2016	<a href="#">Tiago Janela</a>	Initial Version
23 September 2016	<a href="#">Tiago Janela</a>	Updated API spec and requirements
24 September 2016	<a href="#">Tiago Janela</a>	Added special notes regarding the API mock server. Updated the Platform Targets listing. Added a sub-requirement on the Listing screen regarding getting out of search variant. Added SQL Server to the list of databases allowed for back-end implementation.
28 September 2016	<a href="#">Tiago Janela</a>	Completed List Screen requirements.
6 February 2017	<a href="#">Tiago Janela</a>	Added instructions for subject line.
16 February 2017	<a href="#">Tiago Janela</a>	Added instructions for source code and communications language. Underlined and emphasised several points hoping to make them stand out as important.
12 April 2017	<a href="#">Tiago Janela</a>	Updated iOS requirements to allow for Xcode 8.3 and Swift 3.1 submissions
13 April 2017	<a href="#">António Soares</a>	Updated Apiary specification link
13 April 2017	<a href="#">Guilherme Delgado</a>	Updated Platform Targets (Android)
15 July 2017	<a href="#">Tiago Janela</a>	Refined the URL format specification to make a clear distinction between native and web apps.
24 July 2017	<a href="#">Tiago Braz</a>	Refined the specification for Backend developers

## Scope

This document aims to specify the requirements that the software components to be developed must meet in order to be accepted by the final client.

It is comprised of several sections that explore the concepts and data flows needed to meet the acceptance criteria.

Links to the API specification are available in the section [API Specification](#).

**Read the [notes](#) subsections carefully in order to find out what type of application you must build.**

This document also specifies **how the software must be delivered**.

Deliveries made via other mechanisms or that are nonconforming to what is stated in the appropriate section will be reviewed at the discretion of Bliss Applications (**but most likely not accepted**).

Depending on the job position you are applying to, you must develop the software for the appropriate platform (one of the following, **which one is indicated in the [notes](#) subsections** of the [API Specification](#) section):

- iOS
- Android
- Windows (UWP)

- Web Front-end
- Back-end

## Rules of engagement

Any questions can be asked via email to [tianela@blissapplications.com](mailto:tianela@blissapplications.com) or [asoares@blissapplications.com](mailto:asoares@blissapplications.com). We provide no guarantee of response but we will try to answer, naturally, at our earliest convenience during Lisbon timezone working hours or Boston timezone working hours.

**All communication must occur in English.**

## Functional Requirements

### FREQ-01: Loading Screen

The front-end application must show a loading screen while the server health is checked. The server health is checked via the appropriate endpoint on the API.

If the server health is OK then the application should proceed to the “List Screen”.

If the server health is NOT OK then the application should display a “Retry Action” widget.

### FREQ-02: Questions List Screen

The front-end application must show the List Screen in two cases:

- The loading screen managed to contact and check the server health
- The app was opened with an URL with the format
  - For iOS, Android and Windows Apps:
    - `blissrecruitment://questions?question_filter=FILTER`
  - For Web Apps:
    - `http://HOST:PORT/questions?question_filter=FILTER`
  - Notice that this format contains a query parameter which should be used to fill the search box and trigger the search functionality. If the `question_filter` parameter is missing the user should simply be placed at the listing. If the `question_filter` parameter is present but has an empty value the user should be placed at the filter variant with no input inserted but with the input box focused.

The front-end application must fetch list data from the appropriate endpoint taking the following requirements into account:

- The app should fetch 10 records at a time
- There is no sorting functionality. The list will follow the order returned from the API.
- The app should start loading 10 additional records if the user shows intent to browse additional records
- The app should present a search box at the top of the list that allows the user to filter the results. Results should be shown in the same screen as a list. Searching implies hitting the appropriate endpoint and this variation must comply with the 2 requirements defined above.
  - If a search result is being shown (empty or not) the user must be allowed to share this with other users via the “Share Screen”. The app must send an appropriate URL that, when opened, drives the user to the appropriate screen.
  - The app must present a dismiss button to get out of the Search variant.

Each list element is selectable and whenever the user selects one record the app must show the “Detail Screen”.

### FREQ-03: Detail Screen

The front-end application must show the detail screen in two cases:

- A row was selected in “List Screen”
- The application was opened with the detail screen URL from outside the app:
  - For iOS, Android and Windows Apps:
    - `blissrecruitment://questions?question_id=QUESTION_ID`
  - For Web Apps:
    - `http://HOST:PORT/questions?question_id=QUESTION_ID`

The “Detail Screen” must allow the user to navigate back to the listing.

The “Detail Screen” must convey all the information of the object using appropriate visualization widgets.

The “Detail Screen” must allow the user to share this content with other users via the “Share Screen”. The app must send an appropriate URL that, when opened, drives the user to the appropriate screen.

The “Detail Screen” must allow the user to vote on a particular answer via the use of a button. This should trigger an appropriate call to the API endpoint devoted to update Questions

## FREQ-04: Share Screen

This screen must allow the users to share this content with others via email.

The sharing mechanism should invoke the appropriate service on the back-end.

## FREQ-05: No Connectivity Screen

The app must monitor connectivity with the Internet and show an appropriate screen whenever the connection is lost. This screen should remain visible as long as the device has no connection to the Internet. When connection is regained then the user should be at the state where it was before.

# Non-Functional Requirements

## NREQ-01: Platform Targets

- iOS: Objective-C/iOS9 or Swift 2.2/iOS 9/Xcode 7 or Swift 3.1/iOS 9/Xcode 8.3 - carthage or cocoapods
  - Android: Java 7 - API 25 - Android Studio 2.3.1 - gradle 3.3
  - Windows: C#/XAML - UWP (10) - VS 2015 CE
  - Front-end: Javascript/CSS/HTML - vuejs or angularjs - ES5 - Chrome latest version - grunt/npm
  - Back-end: C#, WebAPI - .NET 4.5 - VS 2015 CE - SQL Server or MySQL or Postgres or MongoDB
- We are looking for a .NET MVC application. Not a .NETCore application.

## NREQ-02: Branching Strategies and Configuration Management

The source code must be branched and tagged according to the [gitflow](#) process.

## NREQ-03: Source Code Language

All the source code submitted, included comments, documentation, filenames, project artifact and target names **must be written in English**

## NREQ-04: Software Deliveries

The software will be delivered via the [GitHub](#) platform as a public git repo.

A README file must be present at the root of the repo indicating how to setup, compile and run the software.

You must send an email, **written in English**, to:

[tjanela@blissapplications.com](mailto:tjanela@blissapplications.com) and [asoares@blissapplications.com](mailto:asoares@blissapplications.com)

Indicating the release tag containing the delivery and the subject:

[CANDIDATE\_POSITION RECRUITMENT DELIVERY] {TAG}

In case you are a backend developer and created the tag “v0.1” this would become:

[BACKEND RECRUITMENT DELIVERY] v0.1

# API Specification

Please use the specification available at [apiary.io](#) to implement either the back-end or the front-end.

The base URL for a mock API is:

<https://private-bbbe9-blissrecruitmentapi.apiary-mock.com>

## Note for Mobile, Desktop and Front-end software engineers

This base URL points to apiary’s mock implementation of the API described. This will generate responses for the multiple endpoints but, other than that, it has no real implementation behind. As such, certain endpoints will appear to not be working. That is intended. **Implement the API client against the spec** using one of the

technologies listed on [NREQ-01: Platform Targets](#). That is, partly, what we will evaluate in your case. **Choose a technology that applies to the job position you are applying to.**

## Note for Back-end software engineers

This base URL points to apiary's mock implementation of the API described. This will generate responses for the multiple endpoints but, other than that, it has no real implementation behind. **It is your role to recreate an appropriate API server** that listens on those endpoints and implements the adequate logic, using one of the technologies listed on [NREQ-01: Platform Targets](#). **Choose a technology that applies to the job position you are applying to.**

We **will not provide** a database to link your code to.

You are responsible to setup/populate a Database for your code to run. And your ReadMe should indicate the steps necessary to setup and populate the database on our side in order to test your code against.

If required to run your project, you **must include** any SQL files necessary to make it work.