Ezpark Technology

Proposal, v. 06/11/2019

The Airbnb Of Parking, according to Google Play and Apple App store apps.

1 Quotation

Accomplished work, in hours, **by**
hourly rate between **$32.00** and **$64.00**, up for negotiation, depending on your budget / total amount of hours you are willing to pay for, to achieve your goals, **results in**
total price.

E.g. Minimal hours amount of 256, as an assumption, results in a minimal price of 256 by hourly rate.

1.1 Timeline

Project start date / date of hire / date of proposal submission / **06/11/2019** – **open end**.

E.g. 32 hours per week up to 256 hours, results in a project start date plus 8 weeks timeline.

2 Work Structure

20% of work, to reach the functional requirements / desired quantity, implementing the happy case, following the happy path, working under ideal / narrow set of conditions, with manual help from the developer; **80%** of rework, to reach the non-functional requirements / desired quality, in an iterative manner, visualized as a spiral;

20% of writing in natural language / English and multiple artificial / technical languages; **80%** of rewriting, to reach higher quality;

40% of problem definition; **60%** of problem resolution, whereby the final solution represents its finely detailed problem definition (formulated in natural and artificial languages), being whether right or wrong / true or false, but good or bad / better or worse in reference to the initial state at project start, and having no formal stopping rule / never being perfectly complete (though having natural constraints in form of finite resources for its resolution / time / skill / budget / motivation / state of the market etc.);

80% of thinking, analysis / decomposition, synthesis / composition, (logical) induction, deduction / inference; **20%** of acting / adaptive, well thought out, reacting, as analogies.

2.1 Breakdown

Representing rough approximations, being flexible / adaptive.

2.1.1 Core Processes

Primary, resultative task sequences, adding value to / forming the substance of / targeting the final product / service:

- **10%** planning / requirements analysis and documentation / learning domain specifics / estimation;
- **10%** system and process analysis, structural and object-oriented / process modelling;
- **20%** architecture / design / evaluation of third-party components and services / prototyping;
- **20%** construction / programming;
- **30%** validation / verification / low-level- and high-level-testing / white-box- and black-box-testing / unit-, integration-, system-, load- and acceptance-testing / debugging.

2.1.2 Supportive Processes

Organizational, cooperative tasks, running in parallel, targeting all parties involved, including suppliers of third-party products and services, supporting the core processes by:

- **10%** configuration management / infrastructure configuration / versioning and release-management;
- **20%** project / requirements management / spoken and written communication / mitigation and negotiation / documentation;
- **70%** quality assurance / static and dynamic analysis / build- and run-time measurements.

3 Competence

See attached, rather technical document (Competence.pdf).

4 Platform

4.1 Features

Follow the links in [6.4.3 Web Applications] for details about the (nowadays very large) feature sets of each and every proposed cloud platform / service.

4.2 Benefits

The benefits of cloud computing in general are:

- reduced investment in upfront costs (hardware, software, space, server management); try before you buy (and buy what you really need then);
- flexible pricing (pay based on usage, not for idle capacity);
- centralized configuration / metering / billing / management of multiple services (from the biggest providers; not to confuse with geographically distribution of their data centers);
- flexible capacity; scale up or down based on actual needs (unlimited by the infrastructure, in theory; depending on web application's own implementation, in practice);
- extreme reliability and world-class security (depending on final customer's implementation and configuration, again).

4.3 Lifecycle

The lifecycles of the proposed platforms in general are: date of announcement / launch date / date of commercial availability – today, staying well and healthy, being a bigger player on or even controlling the market / having significant market share, not going to die in the near future.

For timelines of major milestones of the biggest providers (continuously updating their already mature services) see:

- Amazon Web Services (commercially available since 2006);
- Google Cloud Platform (commercially available since 2010);
- Microsoft Azure (commercially available since 2010).

5 High-Level Design

Single, multi-purpose, cloud-based, progressive web application with driver / parking, owner / offering, administrator / management areas, n-factor authentication and role-based authorization.

For details, see [6.3 Frameworks, Languages and Libraries], follow the corresponding links for more details / the sub-components.

6 Proposals

6.1 Programming Methodology

- Structured, imperative, object-oriented (class-based and prototype-based), functional, procedural, generic, reflective, asynchronous / concurrent / parallel, aspect-, event-, task- and test-driven programming,
- with client- and server-side code execution.

6.2 Architectural Styles and Patterns

- Monolithic (deployed as a single unit, not as a collection of interacting services and applications), component-based, database-centric, distributed system;
- with layered / multilayered / multitier / n-tier / three-tier / client-server architecture;
- utilizing model-view-controller, publish-subscribe and cloud computing patterns.

See Architecting Modern Web Applications with ASP.NET Core and Azure for more details.

6.3 Frameworks, Languages and Libraries

Libraries being written in specific languages; frameworks being larger than libraries / compositions of multiple libraries (and other components, e.g. runtimes), written in specific / multiple languages, targeting same / other languages, often being extensions / dialects of another languages.

6.3.1 Server-Side Development

For back-end and front-end development across all layers:

<table>
<tr><td rowspan="2">Web API</td><td colspan="3">Identity</td></tr>
<tr><td>Razor Pages</td><td>MVC</td><td rowspan="2">Entity Framework Core / Dapper</td></tr>
<tr><td colspan="3">ASP.NET Core</td></tr>
<tr><td colspan="4">.NET Core / C#</td></tr>
</table>

and / or

<table>
<tr><td>Telerik UI for ASP.NET Core (based on Kendo UI Professional)</td><td rowspan="2">Braintree Server SDK / SendGrid / Twilio</td></tr>
<tr><td>ASP.NET Core</td></tr>
<tr><td colspan="2">.NET Core / C#</td></tr>
</table>

Integrating components / packages via NuGet.

6.3.2 Client-Side Development

For front-end / presentation layer development:

<table>
<tr><td>Bulma</td><td>Skeleton</td><td rowspan="2">Braintree Client SDK</td></tr>
<tr><td>SASS</td><td>Normalize.css</td></tr>
<tr><td colspan="2">CSS</td><td>JavaScript</td></tr>
</table>

and / or

<table>
<tr><td colspan="6">Bootstrap</td></tr>
<tr><td>SCSS</td><td>Normalize.css</td><td colspan="2" rowspan="2">HTML</td><td>jQuery</td><td>Popvfper.js</td></tr>
<tr><td colspan="2">CSS</td><td colspan="2">JavaScript</td></tr>
</table>

and / or

<table>
<tr><td colspan="3">UIKit / Metro UI</td></tr>
<tr><td>LESS / SCSS</td><td rowspan="2">HTML</td><td rowspan="2">JavaScript</td></tr>
<tr><td>CSS</td></tr>
</table>

and / or

<table>
<tr><td colspan="4">Semantic UI</td></tr>
<tr><td>LESS</td><td>Normalize.css</td><td rowspan="2">HTML</td><td>jQuery</td></tr>
<tr><td colspan="2">CSS</td><td>JavaScript</td></tr>
</table>

and / or

<table>
<tr><td colspan="3">Kendo UI Core / Kendo UI for jQuery (Kendo UI Professional)</td></tr>
<tr><td>LESS</td><td rowspan="2">HTML</td><td>jQuery</td></tr>
<tr><td>CSS</td><td>JavaScript</td></tr>
</table>

or

Angular Material / Kendo UI for Angular (Kendo UI Professional)		
SCSS	Angular	
CSS	HTML	TypeScript
		JavaScript

Angular consuming services from the back-end / via the Web API (s. 6.3.1 Server-Side Development).

6.4 Tools

6.4.1 Desktop Applications

Most actual versions of:

- Adobe Creative Cloud (v. 5.0.0.354, most actual at the time of writing);
- Microsoft Office Professional Plus 2019;
- Microsoft Visio Professional 2019;
- Microsoft Visual Studio Enterprise 2019 (v. 16.3.8) as IDE (for editing, compiling, debugging, profiling, modelling, testing) and Visual Studio Code (v. 1.39.2) as basic / lightweight text editor;
- Skype (v. 8.53 [14.53]) for calling, messaging (depending on functional requirements / personal preferences), screen sharing;
- Upwork Desktop App (v. 5.3.3.800) for messaging, time logging and hourly payment protection, for clients and freelancers (requires, among other requirements, a verified billing method from the client).

6.4.2 Mobile Applications

- Skype for Mobile, for calling, messaging;
- Upwork for Freelancers App / Upwork for Clients App, for formal project management, messaging, notifications.

6.4.3 Web Applications

The online / cloud services from Braintree / PayPal, Google, Microsoft and Upwork:

- Braintree (a PayPal service), for online payments;
- Twilio, as communications-platform-as-a-service (CPaaS) provider:
  - Twilio SMS, for SMS
  - Twilio SendGrid, for Email
- Microsoft Azure (Portal) as infrastructure-, platform- and software-as-a-service (IaaS, PaaS, SaaS):
  - Azure DevOps (Boards, Pipelines, Repos), for functional / technical project management, documentation (above self-explaining, commented code including automated tests);
  - Azure App Service (DNS, Domain [Buy / Migrate], Certificate), as staged (dev / test, production) runtime environment;
  - Azure SQL Database / Azure Cosmos DB (depending on performance requirements / data structures), for data storage;
  - Azure Maps (or Google Maps Platform, depending on functional requirements), for advanced geospatial functionality.

- Upwork Portal, for finance, formal project management, calling / messaging, screen sharing, notifications.

7 Enterprise Architecture

TOGAF and ArchiMate standards, specifications and certifications from The Open Group, BPMN and UML diagrams are not part of this proposal.

8 Product Road Map

- Single product (service) in form of a single progressive web application,
- continually delivered on successful automated builds;
- always live, since initial deployment to product environment.

8.1 Maintenance Period

Infrastructural maintenance (version-control system, runtime environment, database, geo / map services) being done by the platform-as-a-service (PaaS) / software-as-a-service (SaaS) provider of choice (**Microsoft Azure** / Google Cloud / Amazon Web Services).

Quotation does not include costs for these services, billed per use by the provider.

8.1.1 Timeline

Date of initial deployment of the web application to production environment – open end / product (service) lifetime.