

|  |
| --- |
| Homework task3 |
| SAS |

| Related Artifacts | |
| --- | --- |
| Ref. | Name |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Abbreviations and Acronyms | |
|  |  |
|  |  |

Contents

[1 business drivers, goals and objectives 4](#_Toc20499988)

[1.1 list of business drivers, goals and objectives 4](#_Toc20499989)

[2 Stakeholders 5](#_Toc20499990)

[2.1 list of Stakeholders 5](#_Toc20499991)

[2.2 Power/interest matrix 9](#_Toc20499992)

[2.3 RACI matrix 9](#_Toc20499993)

[3 Functional decomposition 9](#_Toc20499994)

# CONTECST diagram



# FUNCTIONAL REQUIREMENTS

## gENERAL REQUIREMENT

The application solution will include functionality available to all members based on permissions, regardless of which module(s) they use. All functionality must be validated against business rules, which will be defined as part of the project. Main customer’s concerns are availability, fault tolerance and recoverability. High level requirements dictate that the system will:

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| **G-1.** | **Allow assignment of user roles** |
| G-1.1. | Each member is assigned a company administrator with the ability to add/edit other users in their company |
| G-1.2. | Company admin can assign roles to users |
| G-1.3. | User roles include Manager, Analytic, Seller, Driver, User; other roles may be added |
| G-1.4. | Support customer-level “Manager” role that has access to manage all data |
| **G-2.** | **Present a role-driven UI based on user role** |
| **G-3.** | **Allow a user’s password to be re-set by the user, the company admin** |
| **G-4.** | **Display a user “dashboard” or start screen, reports and other information** |
| **G-5.** | **Allow users to generate reports** |
| **G-5.1.** | Customize, schedule, run, and view reports for adds, changes, and deletes |
| **G-5.2.** | View usage reports by hour, week, day, month, year |
| **G-5.3.** | Run report data history |
| **G-5.4.** | Run audit reports |
| **G-6.** | **Provide notifications to users in various formats (e.g., email, SMS, and/or onscreen)** |
| **G-6.1.** | Global notifications (e.g. User will be notified of scheduled server down time) |
| **G-6.2.** | Other notifications may be added |
| **G-7.** | **Display contextual help to the user** |
| **G-8.** | **Display history for 1 year** |

## Product REQUIREMENTs

The Product Module is used by Manager, Analytic, Seller to manage orders to the products. High level requirements enable the user to:

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
|  | **Warehouse** |
| **P-1.** | **The list of trade points orders should be presented in Trade Points Orders** |
| P-1.1. | The ability of sort, filter and find for each order should be added |
| P-1.2. | The status for each order should be presented (required, prepared, in delivery, delivered) |
|  | **Analytic/Manager** |
| **P-2.** | **The statistic should be created by client orders and trade point orders** |
| **P-3.** | **The order list with prices should be presented in the Prices window** |
| P-3.1 | The price for each order should be presented |
| P-3.2. | The ability of sort, filter and find for each order should be added |
| P-3.3. | The price aggregation table should be presented |
| **P-4.** | **The order list should be presented in the Orders window** |
| P-4.1. | The status for each order should be presented (ready, in progress, done, refuse) |
| P-4.2. | The ability of sort, filter and find for each order should be added |
| **P-5**. | **As manager I want to know the information of driver locations** |
| **P-6.** | **As manager I want to know the statistic of driver movements, waiting time, gas mileage** |
| **P-7.** | **As manager I want to know the driver ways** |

## trade point requirement

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| **T-1.** | **As Seller I want to see the list of trade points orders** |
| T-1.1. | The ability of sort, filter and find for each order should be added |
| **T-2.1** | **As Seller I want to set up the order status** |
| T-2.1. | The status for each order should be presented (ready, in progress, done, refuse) |
| **T-3.** | **As Seller I want to create order to warehouse** |
| T-3.1. | The order should contain the unit type and unit amount |
| **T-4.** | **As Seller I want to provide/aggregate payment** |

## Mobile application REQUIREMENTs

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| **M-1.** | **As User I want to choose a nearest trade point** |
| **M-2.** | **As User I want to order a beverage** |
| M-2.1. | As User I want to choose a beverage type (coffee, tea) |
| M-2.2. | As User I want to choose a tea/coffee sort |
| M-2.3. | As User I want to choose a milk type |
| M-2.4. | As User I want to choose sugar amount |
| **M-3** | **As User I want to see time for beverage will be ready** |
| **M-4.** | **As User I want to pay for beverage** |
| **M-5.** | **As User I want to add credit card or apple/google pay** |
| M-6. | **As User I want to see my orders history for 1 year** |
| **M-7.** | **As User I want to refuse from beverage** |

## Driver REQUIREMENTs

High level requirements enable the user to:

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| **D-1.** | **As Driver I want to load the itinerary** |
| D-1.1. | As Driver I want to see itinerary |
| **D-2.** | **As Driver I want to start the itinerary following** |
| **D-3.** | **As Driver I want to stop the itinerary following** |
| **D-4.** | **As Driver I want to continue the itinerary following** |
| **D-5.** | **As Driver I want to finish the itinerary following** |
| **D-6.** | **As Driver I want to see a list of unloading stuff** |

# nON-FUNCTIONAL REQUIREMENTS

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
|  | **Mobile application** |
| NFR-1. | Ability to support 10 000 trade points at the same time |
| NFR-2. | Response to Mobile application is not more than 0.5 min |
| NFR-3. | Support iPhone and Adroid platform |
| NFR-4. | Ability to extend system nearest 3 years in different towns |

# Constraints

The solution should be designed to cover following constraints:

* data and business logic must be deployed in order to reach maximum security;
* the architecture should be designed to be “cloud-ready”;
* the architecture should be developed using a service-oriented architecture;

# ASSAMPTIONS

Following assumptions were taken into consideration during elaboration of solution:

* extend mobile application should have ability for QR code usage;
* extend Trade Point should have for ability to create order themselves;
* the solution should be accessible on the following web browsers:

1. Internet Explorer 9 and above;
2. Chrome current and previous version;
3. Firefox current and previous version;
4. Safari current and previous version;
5. Edge current and previous version.

Enterprise Identity Management is external system developed and maintained by Customer.

# Quality Attributes

## Architecturally Significant Quality Attributes

### Utility tree

|  |  |  |  |
| --- | --- | --- | --- |
| **Quality Attribute** | **Quality Attribute Refinement** | **Business/Architecture**  **priority** | **Tactic** |
| ***Availability*** |  | ***H/H*** | Detect faults:   * Monitor – observe both the front-end and back-end system * Sanity check - validation of outputs that provide the mobile devices and trade window (for example sum should be in some range), validate the connection to DB, UI endpoint is available and working * Exception detection – monitor of the system alerts and base of that detects the tactic   Recover from failures:   * Preparation and Repair  1. Passive Redundancy – (a/b deployment) – only one node should process input traffic 2. Exception handling – that is refer with exception detection 3. Software upgrade – (blue/green deployment) - extend functional should be extend, and support all software in actual condition 4. Retry – communication with mobile devices should supported a limit on the number of retries depend on requests (attempts of the payment should be limited, attempts of getting coffee status - not)   Prevent faults:   * Predictive model – the monitoring process can delete the nonperforming process and create a new instance of it * Exception prevention - use of exception classes, use of wrappers to prevent faults * Increase competence set – designing to handle more cases—faults—as part of its normal operation (component that assumes it has access to DB might throw an exception if it discovers that access is blocked) |
| ***Conceptual Integrity*** |  | ***H/M*** | The architecture should be the product of a single architect or a small group of architects with an identified technical leader. This approach gives the architecture its conceptual integrity and technical consistency.   * Frontend, backend are independent in separate modules * Following best practices   Support of the project available page that lists all used libraries and appropriate licenses. New library involvement process has to be discussable beforehand |
| ***Flexibility*** | ***UI/Backend*** | ***L/L*** | Defer Binding tactics allows it’s cost-effective:   * Resource files should be used for configuration * Polymorphism should be used in development as best practice |
| ***Mobile application*** | ***H/H*** | Support mobiles phone types: iPhone and Android  Support Mobile OS latest and 3 years before |
| ***Interoperability*** |  | ***H/M*** | Manage Interfaces:   * Tailor interface – use decorate pattern, develop using interface |
| ***Maintainability*** |  | ***H/H*** | Modifiability:   * Reduce size of module, Increase Cohesion, Reduce Coupling – use in development AOP and Spring as realization * Defer Binding (see in flexibility)   Testability, Reusability(bellow) |
| ***Manageability*** |  | ***H/H*** |  |
| ***Performance*** |  | ***H/H*** | Control resources:   * prioritize events – event rate should be depending on resources,   + max rate is request from customer and trade point   + lower rate is request from warehouse   + lowers rate is request from analytic and managers windows * Increase resource efficiency – use algorithms for development that allow to reduce their processing time   Manage resources:   * Increase resources – use faster processor, additional memory * Introduce concurrency – process the customer requests in parallel * Maintain multiple copies ofcomputations *–* scale the modules that processed customer requests using the load balancer * Schedule resources- scale the modules that processed customer requests depend on the system load |
|  |
| ***Trade point Response Latency = 0.1 – 1 sec*** |
| ***Backend Response Throughout = 10000 request/sec (1000 points)*** |
| ***Backend latency = 0.1 sec*** |
| ***Other windows Latency = 5 sec*** |
| ***Reliability*** |  | ***H/H*** | Fault Tolerance:   * use the second node for hardware redundancy     Recoverability:   * resorting to manual operations   + backend and should be recover automatically   + front-end can be recover manually * arrange a service provider for temporary data processing facilities – use second backend node to process data   Maturity   * the architecture is well defined and communicated to IT staff and business management is established, the process is largely followed   Availability (see above) |
| ***Reusability*** |  | ***M/H*** | Should follow the best development tactics |
| ***Scalability*** |  | ***H/H*** | Horizontal scalability – scale the module that processed the customer requests |
| ***Security*** |  | ***H/H*** | Detect Attacks:   * Detect intrusion – use special tools for it and setup the protected environment from the provider side * Verify message integrity – use the encryption, authenticity   Resist Attack:   * Authorize Actor – use password for all frontend users, and authorized mobile application * Identity Actor – use ID for customers, ID and access code for trade points * Authenticate Actor – all front-end users used the role with limited access to resources * Change default settings – all passwords should be change after fist login,   React to attacks:   * Inform Actor – notify admin/manager about the attack * Revoke access – any system user access should be limited if the virus suspicious occur     Recover from attacks:   * Maintain Audit Trail – support the system log * See availability, Recoverability |
| ***Supportability*** |  | ***H/H*** | * Use system monitoring tools * Use performance monitoring tools * Use common components to provide tracing support in code |
| ***Testability*** |  | ***L/H*** | Control and Observe System State   * Unit testing using the mockup library and setup quality gate rules * Abstract data source - use emulators and testing DB * Sandbox - use for testing before deployment |
| ***Usability*** |  | ***H/L*** | Support User Initiative   * support different browses latest and 3 years earlier * support iPhone and Android mobile OS * support accessibility for people with disability |