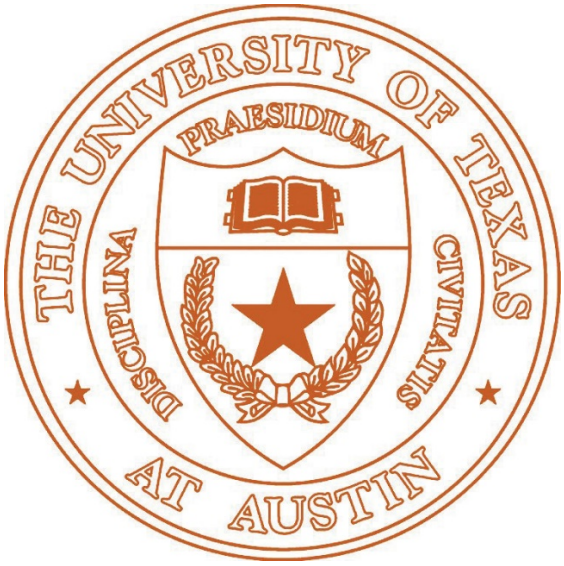
University of Texas at Austin, Cockrell School of Engineering
Software Architecture – EE 382C.7



Assignment # 2
Derivation and Evaluation of Business Blueprints

April 01, 2016

Gabrielson Eapen
EID: EAPENGP

Table of Contents

2. Derivation and Evaluation of Business Blueprints

2.1 Prioritization of Stakeholder Qualities/Constraints along with Quality Categories

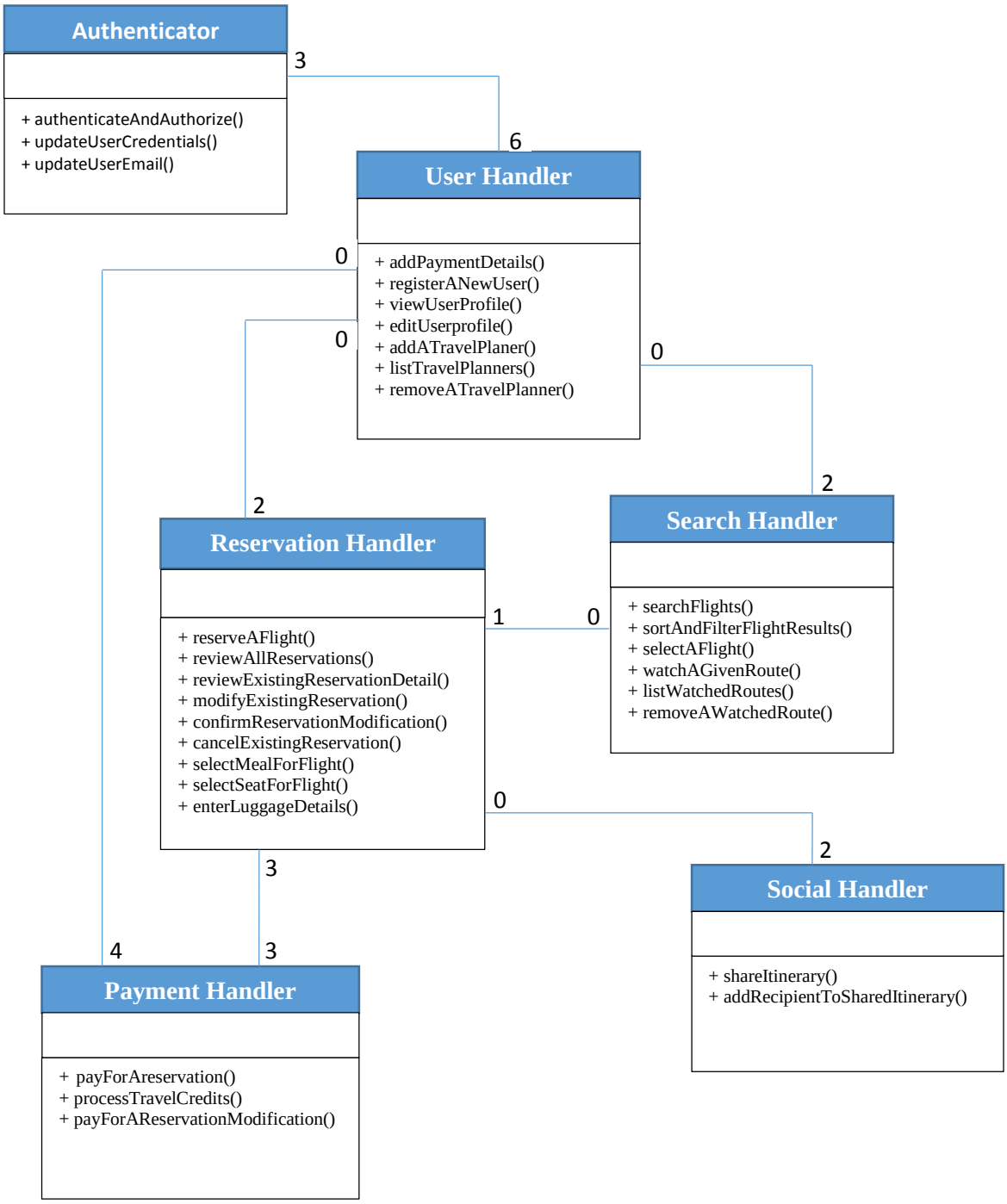
The following is a prioritized table of Stakeholder Qualities and Constraints presented in the problem domain.

Priority	Need/Quality	Classification	Priority Justification
1	The site (application) should be available to end-users for the entire duration of their access (visit). Specifically, the system needs to be available 24x7 with a four 9's (99.99%) uptime goal.	Availability	If travelers are unable to access the site (application) when they need it, they are unable to schedule and plan trips. Besides inconveniencing the traveler, it adversely affects the reputation of the site.
2	The application user interface (UI) should be simplistic and intuitive to use so that end-user needs minimal to no training in order to navigate the site.	Usability	If travelers find it difficult to use or navigate the site, they are less likely to use it. Ease of use leads to higher end-user acceptance and satisfaction.
3	The user should not have to wait for extended periods of time at any point when using the system.	Performance	Since the application is available globally 24x7, end-user usage patterns can vary. But regardless of application load, all application functions must perform consistently and complete in the specified amount of time.
4	Authenticated access to the application should be secure and user credentials should not be compromised. SSL can be used to secure the sensitive information. Sensitive personal or payment information should be encrypted when stored in the database.	Security	The application must comply with industry standards in storage and transmission of sensitive personal information to avoid any legal liability. Specifically collection, transmission, and storage of payment (CREDIT CARD) information or cardholder data must comply with the Payment Card Industry Data Security Standard (PCI DSS).
5	The application should be able to handle large numbers of users and have ability to increase capacity on demand in response to growth or during peak travel seasons.	Scalability	To ensure consistent performance, when user load or activity increased, the application infrastructure needs to be elastic and scale to meet the higher load. This helps satisfy priorities 1 and 3 as well.

6	The application should be easily extensible	Extensibility	As our company desires to agile and aggressive in the market place, it needs to be able to add new features or capabilities with minimal design effort. So tight coupling between modules should be avoided.
7	There should be regularly scheduled backups of critical user and system data to allow for recovery whenever needed.	Data backup and recovery	Hardware failures happen and it would be embarrassing to inform customers that we have lost their data. If possible, continuous or almost real-time backups should also exist to allow restores to a specific point-in-time.
8	The application should be created with proper coding principles and design patterns with controlled releases from development to test to production.	Maintainability	Similar to priority 6, incremental updates and/or bug fixes need to have controlled rollouts (or rollbacks) without availability disruption.
9	Delivery of the application must occur within the agreed upon time frames.	Project Schedule	Our company desires to be agile and aggressive in the market place. However, this is a saturated market and meeting the above goals is more important than schedule.
10	The application should be created within the specified budget without cost overrun.	Project Cost	Adhering to the stipulated budget goes a long way in inspiring confidence in our company's long term viability. But this is the least important as we are a start-up and will take projected costs whatever they may be to the investors.

2.2 Business Blueprint Derivation

2.2.1 Graphical Representation of Components and their Relations



2.2.2 Textual Representation of Components and their Relations

From the given domain, a set of components has been extracted. The section is dedicated to present a textual representation of the mapping between such components with data and functions, as well as the relations and dependencies between components.

2.2.2.1 Allocation of functions and data to components

Component	Data	Functions
Authenticator	User Credentials User Email	authenticateAndAuthorize() updateUserCredentials() updateUserEmail()
Payment Handler	Credit Amount	payForAReservation() processTravelCredits() payForAReservationModification()
Reservation Handler	User Reservation List Trip Reservation Details[1]	reserveAFlight() reviewAllReservations() reviewExistingReservationDetail() modifyExistingReservation() confirmReservationModification() cancelExistingReservation() selectMealForFlight() selectSeatForFlight() enterLuggageDetails()
User Handler	User ID User Profile Information[2] Planner Information[3] Payment Information	addPaymentDetails() registerANewUser() viewUserProfile() editUserprofile() addATravelPlaner() listTravelPlanners() removeATravelPlanner()
Search Handler	Watched Routes	searchFlights() sortAndFilterFlightResults() selectAFlight() watchAGivenRoute() listWatchedRoutes() removeAWatchedRoute()

[1] Trip Reservation details has all the information in the reservation record including trip cost and shared recipients
[2] Profile Information is specified as a single data entity but excludes Email Address
[3] Planner information is specified as a single data entity and includes name and email

Social Handler	Shared Reservation List Reservation Shared Recipients	shareItinerary() addRecipientToSharedItinerary()

2.2.2.2 Function I/O Dependencies between Components

Components	Dependency
FROM: Authenticator TO: User Handler	• "addPaymentDetails" requires "authenticated user id" from "authenticateAndAuthorize" • "viewUserProfile" requires "authenticated user id" from "authenticateAndAuthorize" • "editUserProfile" requires "authenticated user id" from "authenticateAndAuthorize" • "addATravelPlanner" requires "authenticated user id" from "authenticateAndAuthorize" • "listTravelPlanners" requires "authenticated user id" from "authenticateAndAuthorize" • "removeATravelPlanner" requires "authenticated user id" from "authenticateAndAuthorize"
FROM: User Handler TO: Authenticator	• "updateUserCredentials" requires user submitted "username" from "registerANewUser" • "updateUserCredentials" requires user submitted "password" from "registerANewUser" • "updateUserEmail" requires user submitted "email address" from "registerANewUser"
FROM: User Handler TO: Reservation Handler	• "reserveAFlight" requires data "User ID" which was allocated to component "User Handler" • "reviewAllReservations" requires data "User ID" which was allocated to component "User Handler"
FROM: Reservation Handler TO: Social Handler	• "shareItinerary" requires data "Trip Reservation Details" which was allocated to component "Reservation Handler" • "addRecipientToSharedItinerary" requires data "Trip Reservation Details" which was allocated to component "Reservation Handler"
FROM: User Handler TO: Search Handler	• "watchAGivenRoute" requires data "User ID" which was allocated to component "User Handler" • "listAGivenRoute" requires data "User ID" which was allocated to component "User Handler" • "removeAGivenRoute" requires data "User ID" which was allocated to component "User Handler"

FROM: Search Handler TO: Reservation Handler	• "reserveAFlight" requires "flight details" from "selectAFlight"
FROM: Payment Handler TO: Reservation Handler	• "reserveAFlight" requires "paymentSuccess" from "payForAReservation" • "confirmReservationModification" requires "paymentSuccess" from "payForAReservationModification" • "cancelExistingReservation" requires data "Credit Amount" which was allocated to component "Payment Handler"
FROM: Reservation Handler TO: Payment Handler	• "payForAReservation" requires data "Trip Reservation Details" which was allocated to component "Reservation Handler" • "payForAReservationModification" requires data "Trip Reservation Details" which was allocated to component "Reservation Handler" • "processTravelCredits" requires data "Trip Reservation Details" which was allocated to component "Reservation Handler"
FROM: User Handler TO: Payment Handler	• "payForAReservation" requires data "Payment Information" which were allocated to component "User Handler" • "payForAReservation" requires data "User ID" which were allocated to component "User Handler" • "payForAReservationModification" requires data "Payment Information" which were allocated to component "User Handler" • "payForAReservationModification" requires data "User ID" which were allocated to component "User Handler"

2.2.2.3 Function I/O Dependencies between Components and External Producers/Consumers

Components	Dependency
TO: Authenticator	• "authenticateAndAuthorize" requires "Current or New Session ID" from external
FROM: Authenticator	• "authenticateAndAuthorize" sends "Session ID Update Timestamp" from external
TO: Search Handler	• "searchFlights" requires "Airport List" from external • "searchFlights" requires "All Flights Schedules" from external • "searchFlights" requires "Calendar database" from external

FROM: Search Handler	• "watchAGivenRoute" sends "Watched Routes Addition" to external • "removeAGivenRoute" sends "Watched Routes Removal" to external
TO: Reservation Handler	• "reserveAFlight" requires "Passenger Name Record (PNR)" from external
FROM: Reservation Handler	• "reserveAFlight" sends "Passenger Information" and "Itinerary details" for ticketing to external • "confirmReservationModification" sends "Reservation Change Details" to external • "cancelExistingReservation" sends "Reservation Code" to external • "selectMealForFlight" sends "Meal Preferences" to external • "selectSeatlForFlight" sends "Seat Choices" to external • "enterLuggageDetails" sends "Baggage Details" to external
TO: Payment Handler	• "payForAReservation" requires "Authorization Code" from external • "payForAReservationModification" requires "Authorization Code" from external • "processTravelCredits" requires "Confirmed Refund Amount" from external
FROM: Payment Handler	• "payForAReservation" sends "Payment Authorization Request" to external • "payForAReservationModification" sends "Payment Authorization Request" to external

2.2.2.4 Function I/O Satisfied within the Same Component

Components	Dependency
WITHIN: Authenticator	• "authenticateAndAuthorize" references data "User Credentials" allocated to "Authenticator" • "updateUserCredentials" references data "User Credentials" allocated to "Authenticator" • "updateUserEmail" references data "User Email" allocated to "Authenticator"
WITHIN: Payment Handler	• "processTravelCredits" references data "Credit Amount" allocated to "Authenticator"

WITHIN: Reservation Handler	• "reserveAFlight" references data "User reservation List" allocated to "Reservation Handler" • "reserveAFlight" references data "Trip Reservation Details" allocated to "Reservation Handler" • "reviewAllReservations" references data "User reservation List" allocated to "Reservation Handler" • "reviewExistingReservationDetail" references data "Trip Reservation Details" allocated to "Reservation Handler" • "modifyExistingReservation" references data "Trip Reservation Details" allocated to "Reservation Handler" • "cancelExistingReservation" references data "Trip Reservation Details" allocated to "Reservation Handler" • "selectMealForFlight" references data "Trip Reservation Details" allocated to "Reservation Handler" • "selectSeatForFlight" references data "Trip Reservation Details" allocated to "Reservation Handler" • "enterLuggageDetails" references data "Trip Reservation Details" allocated to "Reservation Handler"
WITHIN: User Handler	• "addPaymentDetails" references data "User ID" allocated to "User Handler" • "viewUserProfile" references data "User ID" allocated to "User Handler" • "editUserProfile" references data "User ID" allocated to "User Handler" • "addATravelPlanner" references data "User ID" allocated to "User Handler" • "listTravelPlanners" references data "User ID" allocated to "User Handler" • "removeATravelPlanner" references data "User ID" allocated to "User Handler" • "addPaymentDetails" references data "Payment Information" allocated to "User Handler" • "registerANewUser" references data "User Profile Information" allocated to "User Handler" • "viewUserProfile" references data "User Profile Information" allocated to "User Handler" • "editUserProfile" references data "User Profile Information" allocated to "User Handler" • "addATravelPlanner" references data "Planner Information" allocated to "User Handler" • "addATravelPlanner" references data "Planner Information" allocated to "User Handler" • "listTravelPlanners" references data "Planner Information" allocated to "User Handler" • "removeATravelPlanner" references data "Planner Information" allocated to "User Handler"

WITHIN: Search Handler	• "watchAGivenRoute" references data "Watched Routes" allocated to "Search Handler" • "listWatchedRoutes" references data "Watched Routes" allocated to "Search Handler" • "removeAGivenRoute" references data "Watched Routes" allocated to "Search Handler"
WITHIN: Social Handler	• "shareItinerary" references data "Shared Reservation List" allocated to "Social Handler" • "shareItinerary" references data "Reservation Shared Recipients" allocated to "Social Handler" • "addRecipientToSharedItinerary" references data "Reservation Shared Recipients" allocated to "Social Handler"

2.3 Derivation Plan and Rationale

2.3.1 Derivation Plan

The following table represents all prioritized constraints and the heuristic(s) that can be implemented to achieve needs of the stakeholders. Note that greyed-out (strike-through) heuristics have been eliminated from the derivation as they have been found in direct conflict to a heuristic strategy with a higher priority (for more details, see point 2.3.2).

Priority	Heuristic	Reason	Priority Justification
1. Availability			
1.1	*BB Heuristic:* Group based on Architectural Style – Client/Server based on Functions	*Why:* The client/server architecture has a proven record for highly available and high performance applications. The intent here is to allow grouping of functions that need to be highly available into one or two components	Isolating highly available functions into one component allows for scaling to increase availability when load increases or to provide more resiliency.
2. Usability			
2.1	*BB Heuristic:* Group based on Task Similarity (Similar combination of data and events)	*Why:* The intent is to allow user to access similar function with minimum user input. With the same input/output data, multiple functions can be executed	User does not have to renter data or perform similar actions again and again

2.2	~~*BB Heuristic:* Group based on Similar Capabilities (performer roles)~~	~~*Why:* The intent is to allow user to access similar function associated with the roles they are performing. With the same input/output data, multiple functions associated with the performing role can be executed~~	~~User can stay within a component while performing actions (functions) associated with that role. 2.1 takes precedence when there is a tie.~~
3. Performance			
3.1	*BB Heuristic:* Group based on Data Usage Frequency	*Why:* The intent is to collocate data with the functions that update it to minimize performance overhead by going outside a component	Increasing number of users should not adversely impact performance.
3.2	*BB Heuristic:* Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)	*Why:* Reducing inputs and outputs sent across component boundaries may reduce (i) the likelihood of communication bottlenecks and (ii) the need for inter-component communication channels (most likely slower than intra-component communication)	Most of the data exchanged pertain to the reservation record details. 3.1 should take precedence during a tie.
3.3	~~*BB Heuristic:* Reduce Class Complexity – Size (reduce number of functions in a component)~~	~~*Why:* The intent is to divide the number of the complex functions into the components. So it is easy to extend component with less dependencies.~~	~~Created smaller components that are easier to refactor or repurpose. But it conflicts with the goal of 3.2 and should have lower precedence.~~
4. Security			
4.1	*BB Heuristic:* Store Encrypted Data	*Why:* The intent is to store payment details in encrypted form so it cannot be read without using proper decryption method.	Data can be protected from unexpected and unauthorized exposure as required for compliance with Payment Card Industry Data Security Standard (PCI DSS).
4.2	*BB Heuristic:* Validate User Access	*Why:* Validating user access ensures that only authorized users can access the system. Reservations, Personal Information, or Payment details should not be inadvertently exposed.	System will be secure from the unauthorized users.
5. Scalability			

5.1	*BB Heuristic:* Group based on Data Usage Frequency	*Why:* Grouping based on data usage frequency will ensure minimal performance impact when the system is scaled up	Increasing number of users should not adversely impact performance.
5.2	~~*BB Heuristic:* Group based on Task Usage Frequency~~	~~*Why:* Grouping based on task usage frequency will ensure minimal performance impact when the system is scaled up~~	~~Increasing number of users should not adversely impact performance. During conflict, 5.1 takes precedence.~~
6. Extensibility			
6.1	*BB Heuristic:* Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)	*Why:* Reducing inputs and outputs sent across component boundaries allows for individual component extensibility.	Most of the data exchanged pertain to the reservation record details and User ID. 6.1 should take precedence during a tie.
6.2	~~*BB Heuristic:* Reduce Class Complexity – Size (reduce number of functions in a component)~~	~~*Why:* The intent is to divide the number of the complex functions into the components. So it is easy to extend component with less dependencies.~~	~~Created smaller components that are easier to refactor or repurpose. But it conflicts with the goal of 6.1 and should have lower precedence.~~
7. Data backup and recovery			
7.1	*BB Heuristic:* Plan and Validate infrastructure	*Why:* Planning and validating infrastructure also includes definition of backup and recovery procedures	User personal information and payment details need to be protected.
8. Maintainability			
8.1	*BB Heuristic:* Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)	*Why:* When there is less dependency between components, any fix or enhancement can be done at individual component. It reduces the risk of breaking other components.	It makes bug fixes and enhancements easy. It drives down system cost and requires less time resources.
8.2	~~BB Heuristic: Reduce Class Complexity – Size (reduce number of functions in a component)~~	~~*Why:* The intent is to reduce the number of the functions in the class. So it is easy to find and fix the bug in simpler component.~~	~~While a worthwhile objective, this should not be emphasized at the expense of 8.1.~~
9. Project Schedule			
9.1	*BB Heuristic:* Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)	*Why:* When there is less dependency between components, individual components are easier to implement.	Helps you stay on track and adhere to a proposed schedule.
10. Project Cost			

10.1	*BB Heuristic:* Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)	*Why:* Having less dependency between components, makes implementation easier and keeps cost down. It also reduces the risk of breaking other components.	

2.3.2 Potential Conflicts and Impact on Derivation Plan

Heuristic	Potential Conflict	Possible Relocation
Group based on Similar Capabilities (performer roles) (2.2)	Conflicts with Group based on Task Similarity (2.1) as the heuristic leads to potentially one or two very complex components that map to performer roles	Emphasize Group based on Task Similarity.
Reduce Class Complexity – Size (3.3, 6.2, 8.2)	Conflicts with Reduce Data/Event Dependency (3.2, 6.1, 8.1, 9.1 and 10.1) as the heuristic leads to a higher number of less complex but smaller components that promote	Emphasize Reduce Data/Event Dependency given its usage in Goal #3, 6, 8, 9, and 10.
Group based on Task Usage Frequency (5.2)	May conflict with Reduce Data/Event Dependency (3.2, 6.1, 8.1, 9.1 and 10.1) because the heuristic may attempt to collect functions into larger components.	Emphasize Reduce Data/Event Dependency given its usage in Goal #3, 6, 8, 9, and 10. Cohesion needs to be achieved at an optimum level.

2.4 Evaluate Business Blueprint Structure

2.4.1 Coupling and Cohesion Metrics

2.4.1.1 Number of Inputs/Outputs between components

Component	# Data/Events In	# Data/Events Out	Total
Authenticator	3	6	9
User Handler	6	7	13
Payment Handler	7	3	10
Reservation Handler	6	5	11
Search Handler	2	1	3

Social Handler	2	0	2

2.4.1.2 Number of dependencies between components

Component	# Components to which this component sends or from which this component receives data/events
Authenticator	1
User Handler	4
Payment Handler	2
Reservation Handler	4
Search Handler	2
Social Handler	1

2.4.1.3 Degree of Cohesion

Component	# Functions	# Functions that receive all inputs and send all outputs within component (not counting external)	% Functions that receive all inputs and send all outputs within component (not counting external)
Authenticator	3	0	0%
User Handler	7	1	14%
Payment Handler	3	0	
Reservation Handler	9	6	66%
Search Handler	6	2	33%
Social Handler	2	0	0%

2.4.2 Size and Complexity Metrics

2.4.2.1 Number of functions allocated to a component

Component	# Functions Allocated
Authenticator	3
User Handler	7
Payment Handler	3
Reservation Handler	9
Search Handler	6
Social Handler	2

2.4.2.2 Number of data elements allocated to a component

Component	# Data Elements Allocated
Authenticator	2
User Handler	1
Payment Handler	2
Reservation Handler	4
Search Handler	1
Social Handler	2

2.4.2.3 Number of components in the blueprint

Components in Blueprint
6

2.4.2.4 Component Complexity

Component	# Functions	# Data Elements	# Inputs and Outputs (across all functions)	Complexity
Authenticator	3	2	14	19
User Handler	7	1	32	40
Payment Handler	3	2	16	19
Reservation Handler	9	4	27	40
Search Handler	6	1	14	21
Social Handler	2	2	5	9

2.4.3 Support for Applied Heuristic

Heuristic "Reduce Data/Event Dependency (reduce component-to-component coupling from inputs/outputs)" was selected to help achieve the goals of performance, extensibility, maintainability and to a lesser extent schedule and cost. Considering performance in particular, high value of Data/Event Dependency will result in frequent communication between components in terms of input and output. That will make system slower. In terms of maintainability, high Data/Event Dependency can make bug fix or enhancement more difficult. To accommodate one change we may have to change more than one component and that can increase risk of breaking other components. Same applies to extensibility, extending one component is easy if there is not much dependency. Having low degree of cohesion indicates that the functions will definitely require frequent communication with other components, thereby increasing the latency and interface complexity in the design. It is possible this could eventually impact on overall performance of the system.