Atlanta Regional Commission – MSAA

System Design Document

09/30/2017

Document Number: 10.0

Federal Award ID Number: GA-26-0008-01

Table of Contents

List of Figures

List of Tables

1. Introduction

ARC serves as the Metropolitan Planning Organization (MPO), the Area Agency on Aging (AAA) serving as the Aging and Disability Resource Center (ADRC), the Workforce Board (for a 7-county area) and the Regional Transit Committee (RTC). This structure facilitated collaboration between the AAA and the MPO regarding the need to increase transportation access for older adults and persons with disabilities and the development of the region's first Human Services Transportation (HST) Plan. In 2008, ARC successfully administered the Federal Transit Administration's (FTA) Mobility Services for All Americans (MSAA) grant for a feasibility study for the Atlanta Regional Transportation Management Coordination Center (TMCC). Findings from the 2008 TMCC study supported the development of an HST Advisory Committee and an update of the HST Plan to facilitate greater coordination of HST transportation services throughout the region.

Many Americans have difficulty accessing some of their basic needs, particularly seniors, persons with disabilities and the economically disadvantaged, because they must rely on human service transportation systems which are often fragmented, unreliable, and inefficiently operated. Lack of coordination is the leading obstacle to meeting the mobility needs of the people who need the services most.

In 2015, the MSAA Initiative funded additional deployment planning projects to further improve HST coordination and delivery. The purpose of this deployment planning effort is to replicate and advance the success of TMCC phased-implementation by providing "seed" funding to leverage other federal, state and local resources to build up coordinated community transportation services. MSAA's focus on enhanced coordination supports the realization of USDOT's strategic focus on developing Mobility on Demand (MOD).

Goals are to use service coordination and technology integration to:

- Increase mobility and transportation accessibility for the transportation disadvantaged and the general public.
- Achieve more efficient use of federal transportation funding resources (i.e., do more with less).

Simply Get There Overview

Simply Get There.org is a trip-planning resource for anyone and everyone who lives in or visits metro Atlanta. Users can compare different travel options and costs especially if they needspecialized transportation services. It is a relatively new service developed and hosted by the ARC and its Atlanta Area Agency on Aging (AAA). The web-based application uses a comprehensive listing of public and private sector transportation providers in the Atlanta region to help individuals, especially older adults and persons with disabilities, identify available transportation options. It also provides regional fixed route trip planning options as well as biking and for hire options.

SGT Summary:

- VTCLI one-call, one-click award
- "Trip discovery" tool for public, private, specialized and volunteer transportation services
    - Similar to kayak.com
- Software application developed with Cambridge Systematics
    - Pulls from two ARC-developed databases, ESP and atltransit.org
- Responsive design for use on computers, tablets, and smartphones
- Unique to the Atlanta region
- Includes specialized transportation
- Does not have scheduling capabilities

The project was funded through a Veterans Transportation and Community Living Initiative (VTCLI) grant of the Federal Transportation Administration (FTA) as part of their “One-Click/One-Call” initiative. Launched on March 2015, Simply Get There became the first comprehensive online trip planner for HST populations in the Atlanta region.

1.1 Purpose of the System Design Document (SDD)

The SDD documents and tracks the necessary information required to effectively define architecture and system design in order to give the development team guidance on the architecture of the system to be developed. Design documents are incrementally and iteratively produced during the system development life cycle, based on the particular circumstances of the information technology (IT) project and the system development methodology used for developing the system.

Cambridge Systematics is the original developer of the current solution.

The intent of the solution was to provide a “one call one click” solution. This was not achieved in the initial implementation. The initial vision of the Regional Mobility One-click software application was to link multiple existing call centers to one centralized database with a multi-functional web interface. This concept would maximize staff resources and make transportation information accessible to a wider range of consumers. Transportation resources would be available to the general public and to participating call center operators. Call center staff would have access to a secure client component of the application to register consumers and assist them in accessing/scheduling services. The system would provide an interface for existing client and transportation resource databases from ESP and atltransit.org.

The proposed solution will simply add additional features and functionality to the existing solution to meet the original scope and vision of the Regional Mobility One Click Software application. These features will extend SFT to be a real and integrated one call – one click mobility management solution.

Key points that relate to the design and architecture of the proposed system.

1) No major changes to existing architecture
2) No major changes to system design
3) Major positive impact to the user community
4) Major feature extensions to automate operational functions
5) Major feature extensions to coordinate multiple call centers and transportation providers
6) Regional Coordination API Middleware development

Based on extensive user interviews conducted during the concept of operations phase, it was revealed that the current solution is lacking in key features. It provides strong multi-modal trip planning functions but is limited in back office operational features and lacks one call one click functionality for the consumer. Simply Get There will be expanded to include to create a true one call one click center. The following additional major functionality that users requested to be added or improved include:

- Web-Based Reservations
- Automated Scheduling
- Provider Management
- Trip – Provider Assignment
- Automated Dispatching
- Regional Transportation Coordination
- Regional Cost Allocation

- Automated Fare Payment
- Mobility on Demand Mobile App for consumers and operators

1.2 Audience

The intended audience for the SDD is the project manager, project team, and the future development team. The audience or users for this system design document include the following:

- ARC Project Management Team
- ARC Information Technology Team
- Future application development team
- FTA Project Managers and Oversight Team
- Internal Consulting Team

1.3 Executive Summary

1.3.1 System Overview Summary

ARC has entered a cooperative agreement with FTA to create system specifications for a web-based application that will bring the system forward from “trip discovery” (pinpointing options) to “trip transaction" (centralized booking, scheduling, and dispatching). ARC staff will work with Ride Connection and the third party consultants to design this application. ARC plans to issue an RFP for competitive procurement.

As with websites like kayak.com that aggregate airline data, the long-range vision is for residents to book trips through one online web application, ideally supplemented with phone services. This concept and application design will include the entire process of establishing eligibility, scheduling a trip, finding the right transportation mode and provider, executing the trip, and invoicing the client and paying the provider, as applicable. The application must be designed to be intuitive, supportable, scalable, cost effective, and have the foundation to support future growth. It should also be user-friendly for the general public, transportation and service providers, and ARC staff. ARC has developed an extensive network of external partners and may want to grow or extend the network over time. These partners may wish to access the information directly through a user interface or through an API into their own client system. ARC must have a hierarchy of access points within the application’s administrative functions so that ARC may select the level of access for various external partners.

Project goals include:

1) Integration with Simply Get There trip discovery web application
2) Ability to create client profiles with permissions to use multiple providers, records of current eligibility, trip accommodations needed, and indication of other programs they might join
3) "Trip triaging” capabilities to find ideal cost/accommodations match
4) Ability to schedule a trip
5) Ability to pay for a trip
6) Ability for ARC or a provider to charge a user and for ARC to pay a provider
7) Information on and ability to schedule travel coaching/training assistance
8) Cross-modal trip booking and connections to manifest creation and scheduling systems as well as route optimization across modes
9) Payment and billing - Cost sharing calculated on back-end
10) Data analysis/monitoring to find efficiencies and influence planning/future implementation in a system-wide feedback loop
11) Modular system (“plug and play” system that users could adapt to local needs)
12) Integration with third party systems, including Computer- Aided Dispatched /Automatic Vehicle CAD AVL software, Google, Google Maps, RouteMatch, and Trapeze
13) Ability to track trips by the funding source
14) Ability to generate invoices

15) Web-based application that can be hosted or deployed locally on ARC servers or a location of ARC's choosing
16) A robust API to map data from other ARC and partner systems
17) Ability to house some transportation provider information on this application, rather than pulling all information from two external databases
18) Ability to be 508 compliant

The major new functional modules and extensions to support the goals above may include:

- Coordinated Eligibility Determination
- Coordinated Resource Management
- Automated Web Reservations
- Electronic Payment
- Automated Scheduling and Provider Assignment
- Route Planning and Optimization
- Multi Modal Transportation Coordination
- Real Time Vehicle Tracking and Dispatching
- Transportation Verification
- Transportation Data Analytics
- Customer Mobile App
- Driver Mobile App
- Regional Coordination API Middleware

ARC requires a design with specific functionality to model internal business processes, workflows, partner needs, and integration requirements. ARC requires a design that allows ARC to own the application but may become open source that can be available for use in other parts of the U.S.

User Types

SGT is designed to serve the needs of many different types of users, with features and functions appropriate for each one:

- Travelers are individuals in need of transportation services. Registered Travelers have a user account and travel profile, while anonymous Travelers do not have an account and can use the system without logging in.
- Buddies are friends, family members, or other caregivers who assist Travelers in creating trip plans and managing account settings.
- Agents are customer service representatives who assist Travelers in creating trip plans and managing account settings.
- Agency Administrators are the managers of Agents, who perform maintenance functions related to their Agency.
- Provider Administrators are representatives of organizations that provide transportation services, who need to manage and maintain information on the services they provide.
- System Administrators are the "super-users" who manage the SGT software.

Modes Currently Supported in SGT

- Bicycle;
- Drive;
- Paratransit from local providers;
- Taxi;
- Transit (Bus, Rail) based on General Transit Feed Specification; and,
- Walk;
- UberX

1.3.2 Design Constraints

The proposed solution will utilize the current architecture and system design of the current solution. The current solution is hosted in an industry leading application hosting and data center. Performance, storage, security, and access can be easily scaled using to meet the minimal amount of additional resources the proposed solution will require. This will be a financial constraint that must be considered.

Financial

The largest design constraint for the implementation of the project is financial. The full implementation of the project could be financially significant. ARC intends to implement a phased approach to manage this constraint.

Technical

The development and integration of the new software components into the existing open source software application is a major constraint. Specifc skills and technical understanding of mobility management and demand response management and optimization will be required. This knowledge and skillset is very specific and narrow. Detailed business requirements and use cases will assist in minimizing this challenge.

Transportation coordination and trip sharing will be a major technical consideration. The proposed system must support regional coordination features and provide the ability to integrate trip data into other scheduling and dispatching systems. The coordination function must allow for easy integration and provide open published API's.

Due to the fact that the application is currently hosted and managed by ARC staff, we do not envision any technical computer hardware, network, internet, or database maintenance challenges.

Institutional

The proposed system will be utilized by multiple third party agencies and organizations. This will require coordination and collaboration across the region. Stakeholders that currently have automated scheduling systems may have to integrate into the proposed system via a "reginonal trip coordination" API or comparable solution.

1.3.3 Future Contingencies

The current application has multiple third party dependencies. These third party dependencies are mission critical to the application. Failures or service stoppage severely impacts the applications capabilities.

SGT utilizes existing third party dependencies. These are currently availalble and published. The dependencies include:

- Google Maps API – Utilized as mapping and geocoding engine for the application.
- Enhanced Services Program (ESP) Database Connector – Serves as data source for HHS and demand response service providers.
- OpenTrip Planner API – Utilized to calculate fixed route trip itinerary.

Fixed Route Trip Planning API

The fixed route trip planning functionality utilizes Open Trip Planner (OTP). If OTP becomes unavailable or the service stops, SGT will fail. Google Maps would be a viable alternative for trip planning functionality. OTP is open source which would allow ARC to maintain the service themselves. This would require a minimal level of effort to maintain and manage.

Demand Response Options API

The current application incorporates demand response data from an in-house custom application – ESP. ESP is maintained and managed by the Aging Resources staff and utilize the system for information and referral. SGT is dependent on the transportation provider database. If not available, an alternative source would need to be developed, purchased, or integrated. This would be a major effort and not a good alternative. The risk of ESP becoming unavailable is minimized. ARC owns and manages this application directly.

Due to the lack of major architectural and system design changes associated with the proposed solution, contingency risks are very minimal.

1.3.4 Document Organization

This document completely describes the system at the architecture level, including subsystems and their services, hardware mapping, data management, access control, global software control structure, and boundary conditions. The document is organized into nine major sections. Each section provides detailed sub-sections relevant to the major section. Charts, tables, and graphics have been inserted to explain or clarify content.

2. General Overview and Design Guidelines/Approach

2.1 General Overview

The current solution was built by Cambridge Systematics through funding from the Federal Transit Administration's Veterans Transportation and Community Living Initiative grant program. ARC has privately labeled this application Simply Get There (SGT). SGT is an open source mobility management and cross-modal trip planning software that connects people who need transportation to education, work, health care, and other vital services in their communities.

SGT is built on an open source framework. Source code is available using standard open source management tools such as Git. All source code is stored in a Github repository. Any developer can contribute to the application.

SGT utilizes an open-source web server, Nginx. Nginx is a free open source web server. Nginx is focused on high performance, high concurrency and low memory usage. Additional features on top of the web server functionality, like load balancing, caching, access and bandwidth control, and the ability to integrate efficiently with a variety of applications, have helped to make Nginx a good choice for modern website architectures. Currently Nginx is the second most popular open source web server on the Internet.

SGT is hosted on Heroku. Heroku is a cloud Platform-as-a-Service (PaaS) supporting several programming languages that is used as a web application deployment model. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports multiple other languages.

SGT's development language is Ruby. Ruby is commonly integrated with Rails, a software library that extends Ruby's capabilities. This framework is commonly referred to as "Ruby on Rails". Software developers must be familiar with this framework in order to maintain or build additional functionality into the application.

SGT's database is Postgres. Postgres is also open source and it is very popular and utilized across many open source applications. Postgres is an object-relational database (ORDBMS). It has an emphasis on extensibility and standards compliance.

Github is used as the software development platform. Github provides version control and source code management. Github is the largest host of soure code in the world.

In summary, the existing system design includes the following sub-systems:

- SGT Web Application
- Postgres Database
- Ngenx Web Server
- Heroku Development Platform
- Heroku
- Github Version and Source Code Control

There is no expectation that any of these systems will be changed or modified with the proposed system.

2.2 Current

A Statement of Need explains why the system is being developed, what purpose it serves, and why it is necessary.

SGT was designed to meet the transportation needs of human service transportation clients such as Veterans, military families, elderly, disabled, other transportation disadvantaged.

SGT is a trip planning system designed to meet the transportation needs of human service clients including veterans, military families, elderly, disabled and other transportation disadvantaged groups. It:

- Provides unified trip planning for public, private and volunteer services;
- Works on computers, tablets, and smartphones;
- Is tailored to an individuals' trip planning needs; and
- Empowers call center staff to deliver improved services.

Veterans and their families are often in need of transportation services to enable them to attend medical appointments, receive physical therapy or mental health counseling, seek jobs or education, and access various veterans and related community services. Some veterans and family members have disabilities – mental, physical, or developmental – that exacerbate the challenge of obtaining these services. Numerous other populations experience similar challenges, including the elderly, transit-dependent populations, and other nonveterans with disabilities.

SGT enables these target populations to quickly and easily identify the most appropriate options for making a particular trip, evaluating and identifying options that include fixed-route transit, demand-responsive transit (DRT), taxi and other private transportation services, paratransit, volunteer transportation service networks, carpools, and vanpools. 1-Click also provides call center or social service agency staff with a single, centralized source of this information to use on behalf of their clients.

SGT tailors trip plan options to the needs, preferences, and schedules of each individual, based on factors such as Medicaid eligibility, veterans' transportation eligibility (which depends on Veteran status and trip purpose), age, physical mobility limitations, and other preferences regarding tradeoffs of time, cost, and convenience.

SGT also stores data that can be used to generate a variety of reports on system usage, mobility impacts, trips planned and made, and unmet transportation needs.

2.2.1 Proposed Solution - Statement of Need

The current solution does not provide call center operational support nor does it provide the ability to coordinate other regional call centers and transportation resources. The solution must be extended to support these functional needs. The proposed system will dramatically improve call center operations, regional coordination, and, most importantly, the customer experience. Customers will be able to plan and reserve transportation online. Transportation providers will be able to connect to the one click system in real time to schedule the trip. Operations will have the ability to assign trips and to monitor performance in real time. Automated scheduling and routing will be available to optimize transportation resources. Automate dispatching tools will be available to the call center and to providers. This will create a single coordinated system for HST and demand response transportation. Transportation Network Companies (TNC) will also be integrated into the solution for a true multi-modal application. Transportation data analytics will be available at a regional level. Ultimately, the proposed system will execute the intended vision and requirements of a "one click" mobility management solution.

2.3 Stakeholder Roles/Responsibilities/Concerns

System design can cross many different groups within an organization to ensure requirements are gathered and met for all stakeholders. As such, the roles and responsibilities section may be necessary to provide the team with clarification on who performs various roles. This section also serves as a list of points of contact for the team and stakeholders should issues and concerns arise which need to be addressed.

Regional Stakeholders

1. ARC Aging and Disability Resource Connection (ADRC)
2. ARC Transportation Access and Mobility Services Division

3. ARC Area Agency on Aging
4. Center for Visually Impaired (CVI)
5. City of Atlanta, Vehicles for Hire/Taxi Management
6. Cobb Community Transit (CCT)
7. DeKalb Office of Senior Affairs
8. Disability Link, the Center for Independent Living (CIL)
9. Goodwill Industries
10. Georgia Commute Options (GCO)
11. Georgia Department of Community Health (DCH)
12. Georgia Department of Human Services (DHS)
13. Georgia Department of Transportation (GDOT)
14. Georgia Governor's Development Council (GDC), Rural and Human Services Transportation (RHST)
15. Georgia Transit Association (GTA)
16. Gwinnett County Senior Services
17. Lifespan Resources (volunteer driver program)
18. Metropolitan Atlanta Rapid Transit Authority (MARTA)
19. Ride Connection of Portland, Oregon
20. Atlanta United Way 211
21. Veterans Affairs (VA), Veterans Transportation Program (VTP)

Additional support is provided by Kevin Chambers, IT Director of Ride Connection in Portland, OR.

Technical / Project Stakeholders

The following table provides the role and contact information for the key technical and project stakeholders associated with the system design.

Table 1: Project members contact information

Name	Role	Email
Mary Blumberg	Executive Sponsor	mblumberg@atlantaregional .com
Cynthia Burke	Project Manager	Cburke2@atlantaregional.com
Leslie Caceda	Application Owner	lcaceda@atlantaregional.com
Ray Randolph	IT Director	rrandolph@atlantaregional.com
Tim Quinn	Technical Lead	Tim.quinn@thingtech.com
Carly Harper	Business Consultant	Carly.harper@thingtech.com

2.3.1 Roles

SGT is designed to serve the needs of many different types of users, with features and functions appropriate for each one:

- Travelers are individuals in need of transportation services. Registered Travelers have a user account and travel profile, while anonymous Travelers do not have an account and can use the system without logging in.
- Buddies are friends, family members, or other caregivers who assist Travelers in creating trip plans and managing account settings.
- Agents are customer service representatives who assist Travelers in creating trip plans and managing account settings.

- Agency Administrators are the managers of Agents, who perform maintenance functions related to their Agency.
- Provider Administrators are representatives of organizations that provide transportation services, who need to manage and maintain information on the services they provide.
- System Administrators are the “super-users” who manage the SGT software.

	General	Traveler	Agent	Agency Administrator	Provider Administrator	System Administrator
Plan Trips	✓	✓	✓	✓	✓	✓
Maintain Profile and Places		✓	✓	✓	✓	✓
Create Traveler Accounts			✓			✓
Assist Travelers			✓			
Create Agencies						✓
Maintain Agencies				✓		
Create Providers						✓
Maintain Providers					✓	
Create/Maintain Services					✓	
Send User Messages						✓

Figure 1: SGT Roles

The following table identifies the system design roles. This matrix also serves as the list of points of contact for issues and concerns relating to the system design.

Table 2: System design roles

Name	Role	Phone	Email
Not Identified	Project Manager		
Not Identified	Lead Designer – User Interface		
Not Identified	System Architect		
Not Identified	Software Developer		
Not Identified	Quality Assurance Lead		

2.3.2 Responsibilities

Development team has not been selected to design and build the extended functionality to the current system. However, the items below define the roles for the project.

Project Manager

Project management responsibilities include delivering every project on time within budget and scope. Project managers should have a background in business skills, management, budgeting and analysis.

Responsibilities:

- Coordinate internal resources and third parties/vendors for the flawless execution of projects
- Ensure that all projects are delivered on-time, within scope and within budget
- Developing project scopes and objectives, involving all relevant stakeholders and ensuring technical feasibility
- Ensure resource availability and allocation
- Develop a detailed project plan to track progress
- Use appropriate verification techniques to manage changes in project scope, schedule and costs
- Measure project performance using appropriate systems, tools and techniques
- Report and escalate to management as needed
- Manage the relationship with the client and all stakeholders
- Perform risk management to minimize project risks
- Establish and maintain relationships with third parties/vendors
- Create and maintain comprehensive project documentation

Lead Designer – User Interface

UI designer is responsible for creating intuitive user experiences. The ideal candidate should have an eye for clean and artful design, possess superior UI skills and be able to translate high-level requirements into interaction flows and artifacts, and transform them into beautiful, intuitive, and functional user interfaces.

Responsibilities

- Collaborate with product management and engineering to define and implement innovative solutions for the product direction, visuals and experience
- Execute all visual design stages from concept to final hand-off to engineering
- Conceptualize original ideas that bring simplicity and user friendliness to complex design roadblocks
- Create wireframes, storyboards, user flows, process flows and site maps to effectively communicate interaction and design ideas
- Present and defend designs and key milestone deliverables to peers and executive level stakeholders
- Conduct user research and evaluate user feedback
- Establish and promote design guidelines, best practices and standards

Software Architect

Responsible for initial design and development of new software or extensive software revisions; products may be for use internally or for resale. Defines product requirements and creates high-level architectural specifications, ensuring feasibility, functionality, and integration with existing systems/platforms. Requires a bachelor's degree and may be expected to have an advanced degree in area of specialty and at least 7 years of experience in the field or in a related area.

- Demonstrates expertise in a variety of the field's concepts, practices, and procedures.

- Relies on extensive experience and judgment to plan and accomplish goals.
- Performs a variety of complicated tasks.
- May provide consultation on complex projects and is considered to be the top level contributor/specialist.
- May guide a team of developers through the project to completion. Typically reports to a head of a unit/department or top management.

Software Engineer

The software engineer builds high-quality, innovative and fully performing software in compliance with coding standards and technical design. Software engineer responsibilities will include development, writing code, and documenting functionality.

- Execute full lifecycle software development
- Write well designed, testable, efficient code
- Produce specifications and determine operational feasibility
- Integrate software components into a fully functional software system
- Develop software verification plans and quality assurance procedures
- Document and maintain software functionality
- Tailor and deploy software tools, processes and metrics
- Serve as a subject matter expert
- Comply with project plans and industry standards

Quality Assurance Lead

QA engineer responsibilities include designing and implementing tests, debugging and defining corrective actions. You will also review system requirements and track quality assurance metrics (e.g. defect densities and open defect counts.)

Responsibilities

- Review requirements, specifications and technical design documents to provide timely and meaningful feedback
- Create detailed, comprehensive and well-structured test plans and test cases
- Estimate, prioritize, plan and coordinate testing activities
- Design, develop and execute automation scripts using open source tools
- Identify, record, document thoroughly and track bugs
- Perform thorough regression testing when bugs are resolved
- Develop and apply testing processes for new and existing products to meet client needs
- Liaise with internal teams (e.g. developers and product managers) to identify system requirements
- Monitor debugging process results
- Investigate the causes of non-conforming software and train users to implement solutions
- Track quality assurance metrics, like defect densities and open defect counts
- Stay up-to-date with new testing tools and test strategies

2.3.3 Concerns

Due to the fact that the proposed system is simply additional features and will not require any design or architectural changes, there are no technical concerns.

2.4 System Assumptions/Constraints/Dependencies/Risks

2.4.1 Assumptions

The largest assumption is that the existing SGT trip planning application will be extended to support the proposed new features. The existing architecture and system design will be used including all existing components and sub-systems. It is certain that additional functionality will be added to the proposed solution.

2.4.2 Constraints

There are no hardware, software, or software technical constraints identified with this project. Financial constraints are a potential constraint since funding has not been identified to build the proposed solution. Institutional constraints mays exist due to the systems need for regional coordination, participation, and interoperability.

2.4.3 Dependencies

The current application is dependent on many third party systems. These include:

- Open Trip Planner
- Google Maps
- ESP
- Uber (if shared ride mode is enabled)

The current application is also dependent on accurate GTFS data. ARC is currently responsible for maintaining the GTFS data for the regional fixed route providers.

2.4.4 Risks

Minimal risk is associated with the system design. This is primarily due to the fact that the existing system design and architecture will not be modified to meet the needs of the proposed solution. Financial risks are a concern. Funds have not been identified to fund the proposed project. Ongoing maintenance of the system will also be a concern.

Alignment with National/Regional ITS ArchitecturesThe current and proposed solution aligns with the National and Regional ITS architecture. The proposed solution, if implemented, will adhere to all appropriate federal ITS architecture mandates.

Design Considerations

SGT is a trip planning system designed to meet the transportation needs of human service clients including veterans, military families, elderly, disabled and other transportation disadvantaged groups. It:

- Provides unified trip planning for public, private and volunteer services;
- Works on computers, tablets, and smartphones;
- Is tailored to an individuals' trip planning needs; and
- Empowers call center staff to deliver improved services.

The major design considerations for the proposed extended features are related to system performance and scalability of the solution. Data center is hosted in AWS which provides a tremendous amount of flexibility in terms of scaling the performance. Processor speed, memory, peripherals, and stakeholder support will be factored in the design.

2.5 Goals and Guidelines

The following goals must be addressed in the execution of the proposed solution.

Leverage Existing Architecture

The proposed solution must leverage the current architecture and system design used by current solution. This minimizes negative impacts on usability, user experience, and financials. The proposed solution will simply extend the current application to support additional features, functionality, and use cases.

Development Environment

The application development environment must remain consistent. This minimizes negative impacts to interoperability and quality. ARC does not wish to re-write or re-engineer the existing application unless absolutely necessary.

Ease of Use

The new features must be easy to use and provide a strong user experience. New features cannot impact existing functionality from a user perspective.

Extensibility

The proposed features must be extensible. Features can be enabled as needed or required by the users.

API Enabled

Regional coordination support is a key driver of the project. The application must be API centric and support an open and published API architecture.

RESTful Framework

The application and underlying architecture must be a REST framework.

2.6 Operational Environment

- Ruby on Rails Development
- Git Version Control
- Github Repository
- PostgreSQL Database
- Apache Web Server
- NGINX Server

Functional goals of the proposed system includes:

- Extending functionality of the existing web application
- Improving application performance
- Sharing and coordinating data via a distributed model
- Completing the one-call one click deployment model

2.7 Development Methods & Contingencies

The basics of a good architecture is to layer the application into multiple autocratic and autonomous applications that can be replaced individually and allow us to keep the application running while we are working on a specific layer. The communication between each layer should be a RESTful API call with JSON content.

Scalability

Ensure that the architecture can be scaled horizontally, across multiple servers and across multiple regions. That means that once your traffic goes up, you should be able to add and remove new servers as the solution requires.

Availability

The architecture should support a high availability environment. Infrastructure redundancy is required. This ensures the solution is available if multiple servers or an entire data center fail. The current availaibility of the solution per the hosting providers service level agreement is 99.999% availability.

Security

Solution architecture should expose only the minimal amount of code possible. Most of the back-end pieces should be hidden away. In addition to that, security of each system should be multi-layered.

Extensibility

Architecture must be able to swap out modules, change layers, and add pieces to the application without having to worry about the underlying data contracts in place.

Separation of responsibility

System should be modular enough that each piece of code has a set of responsibilities and not more. The back-end should not create front end code nor should the front-end code include business logic.

RESTful Framework

The reason for a RESTful API is plain and simple flexibility. Framework does not want to be tied or dependent on a specific programming language and architecture (Java or C#). Architecture needs to be able to replace each layer independently and even use different languages that might be better suited for a certain layer.

2.8 Architectural Strategies

The Cloud trend is one of the most disruptive and challenging forces impacting customers' applications and infrastructure, requiring new business models and new architecture decisions, which impact how organizations deploy, manage, maintain, and protect and manage their data.

Amazon Web Services offers multiple options for provisioning IT infrastructure and the deployment of web-based applications. The deployment model varies from customer to customer. Below are the key strategies associated with this model.

Infrastructure On Demand

In a non-cloud environment: (i) infrastructure assets require manually configured, (ii) capacity requires manual tracking, (iii) capacity predictions are based on the guess of a theoretical maximum peak, and (iv) deployment can take weeks. Within the cloud, these building blocks that represent the Infrastructure are not only provisioned as required, following actual demand and allowing pay-as-you-go, but can also be programmed and addressed by code. This greatly enhances flexibility for both Production/Dev/Test environments as well as Disaster Recovery scenarios. Resources can be provisioned as temporary, disposable units, freeing users from the inflexibility and constraints of a fixed and finite IT infrastructure.

Infrastructure can be automated through code, allowing for greater self-service and more automated delivery of desired business and technical outcomes. Consumption is measured by what you consume, not what you could consume, drastically changing the DR cost modelling challenges experienced today. This represents a major, disruptive reset for the way in which you approach Disaster Recovery, testing, reliability and capacity planning.

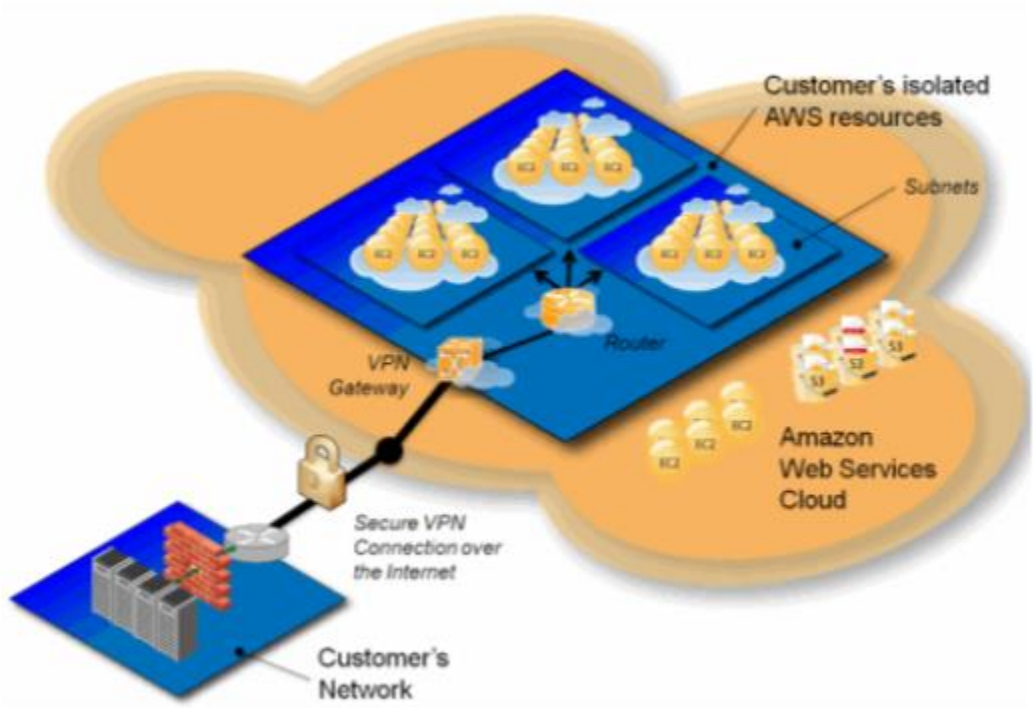


Figure 2: Amazon Web Services model

Cloud Computing

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing has become the primary engine driving IT as a service. With cloud computing, you don't need to make large upfront investments in hardware and spend a lot of time managing that hardware. Instead, you can provision exactly the right type and size of computing resources you need to power your newest bright idea or operate your IT department. As the cloud has become mainstream and adoption has garnered momentum, you have access to state-of-the-art technology at a fraction of the cost and with greater speed than ever before.

AWS offers global infrastructure available to Customers on a pay-as-you-go model, allowing for more flexibility in meeting requirements for Data Protection and Disaster Recovery. Resources, bandwidth and their availability can now be localized to your corporate assets and human resources, allowing for a more distributed footprint that reduces backup windows and simplifies data protection that otherwise would be cost prohibitive with a physical datacenter or co-located approach, all while maintaining a simplified, unified pay-as-you-go billing approach.

Disaster Recovery

Physical DR environments have less capacity than their Production, or Dev/Test counterparts, resulting in degraded service in the event of a failover. Even more so, hardware is often re-purposed to fulfill the DR environment's requirements, resulting in higher than expected maintenance costs. With the Public Cloud model, this hardware availability and refresh aspect is disrupted by removing the need to maintain a hardware fleet that can meet both your DR requirements and sustain your service level agreements. You can provision instances to meet your needs, when you need them, and for specific DR events – both real and test – and the underpinning hardware is maintained and upgraded by the Cloud provider without any need for technical input, and no upgrade costs are incurred by the organization. This dynamic shift allows you to begin costing per DR event, instead of paying for availability, improving your level of Disaster Recovery Preparedness through the application of flexible, unlimited resources to stage both DR tests and execute actual DR events.

Mobility

Mobility has fundamentally changed the way businesses operate. Information is available in multiple devices, in real time, and with greater accuracy than ever before. Mobility and the cloud together make it easier for workers to be productive from anywhere—not just the office. The mobile tools they use must be secured to meet tough industry standards.

Social

Social media has had a significant impact on the way people work. Employees can share information in real time, with multiple inputs and transparency.

In a non-cloud environment you would have to provision capacity based on a guess of a theoretical maximum peak. This can result in periods where expensive resources are idle or occasions of insufficient capacity.

Scalability

Applications grow over time, and a Data Management solution needs to adapt with the change rate to protect the dataset quickly and efficiently, while maintaining an economy of scale that continues to generate business value out of that system.

Backup/Archive to the Cloud

Protecting data at the primary on-premise location by writing directly to an external cloud provider's storage solution, or retaining a local copy and replicating the backup/archive data (either in full, or only selective portions of that data) into an external cloud provider's storage service

Platform

Platform as a Service (PaaS) is the next step down from Software as a Service (SaaS) in the Cloud Computing Stack.

PaaS provides the platform for developing SaaS applications and services.

Includes software development tools, network connectivity, application servers, database management, enterprise service buses, analytics, etc…

- OpenShift
- Heroku
- Amazon

The current application utilizes a platform as a service (PaaS). Below is the summary of this service.

- Heroku Platform as a Service (PaaS): Cedar -14 Stack using Ubuntu 14.04 Linux as a basis
- Polyglot Platform – native support for development with :
    - Ruby or Rails
    - Node.js, Angular
    - Java, Spring or Play
    - Python or Django
    - Clojure

    - Scala
- Process Model with OS Kernel, Web Server Configured on Dyno Spin-Up

Development Environment

Table 3: Software descriptions

Software	Description
Github	Version control repository
Heroku / Amazon Web Services	Cloud computing platform
Force.com	Cloud computing platform
Postgres SQL	Database
Bootstrap	UI Framework / Theme
Node / Node.js	Programing Language
ReactJS	Programming Framework for Web UI
React Native	Programming Framework for Mobile
APEX	Force.com Programming Language

Open API Centric

APIs allow for the creation of a minimal interface that is relatively stable that can be used by other software systems to access or manipulate the underlying systems or data. This allows for enhancements to the underlying systems or data without disturbing the software systems that use the API Usually implemented using REST, SOAP, or JSON. Third party application and database integration is simplified as long as all parties support the published API.

2.9 Performance Engineering

AWS provides multiple options to configure and procure related services to eliminate potential performance issues.

3. System Architecture and Architecture Design

This section outlines the system and hardware architecture design of the system.

3.1 System Architecture Diagrams

This section provides the conceptual view of the system and its functionality.

Simply Get There currently provides the following major components.

- Trip Discovery
- Eligibility
- Trip Review
- Trip Plans

3.1.1 External Systems diagram

Instructions: Provide an external systems diagram model of the interaction of the system with other external systems in the relevant contexts, thus providing a definition of the system's boundary in terms of the system's inputs and outputs.

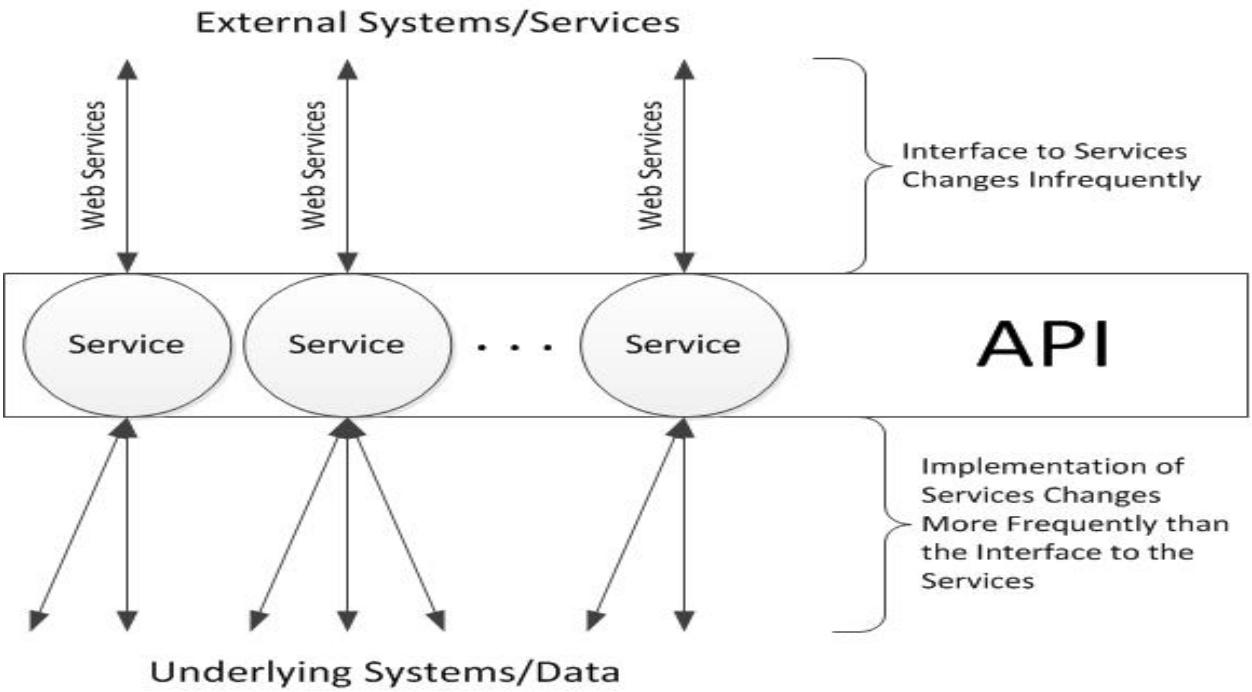


Figure 3: External systems diagram

3.1.2 Functional/Logical diagram

Instructions: Insert any related functional/logical views or provide a reference to where they are stored. A functional architecture is a logical model of the functional decomposition. The logic model provides a depiction of the flow of inputs and outputs and it provides a tracing of inputs and output to specific functions and items representing the system.

3.2 Hardware Architecture

Different deployments can use different server configurations, but SGT is typically deployed on four servers:

1. A web server running Apache to host the web application;
2. Storage for various application configuration files (e.g., CSS, images, etc.);
3. An OpenTripPlanner server to respond to trip planning requests; and
4. A PostgreSQL database server to host the 1-Click database.

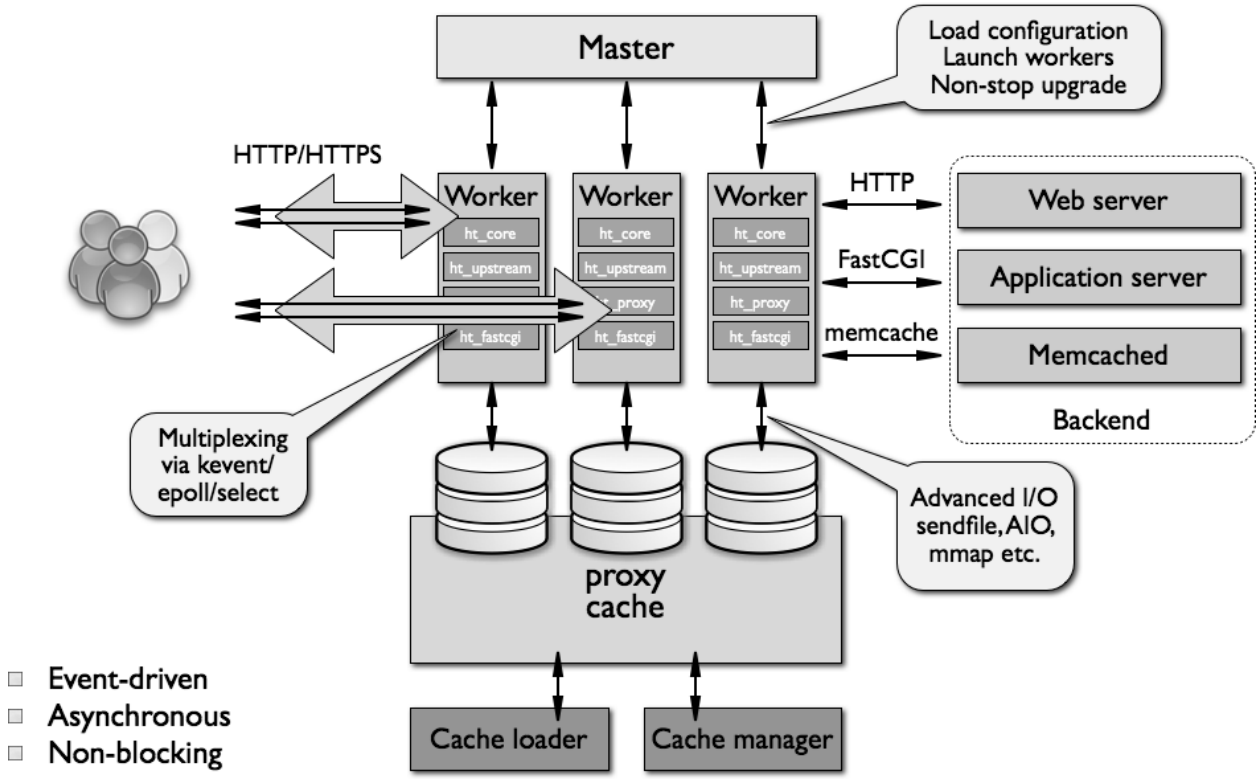


Figure 4: Hardware architecture

Table 4: Systems descriptions

	SYSTEM	NOTE
PROGRAMMING LANGUAGE	Ruby	Ruby is a programming language. It was created 20 years ago by Yukihiro “Matz” Matsumoto. By most measures of programming language popularity, Ruby ranks among the top ten, though usually as tenth (or so) in popularity, and largely due to the popularity of Rails. Like Java or the C language, Ruby is a general-purpose programming language, though it is best known for its use in web programming.

APPLICATION FRAMEWORK	Rails	Rails is a software library that extends the Ruby programming language. Rails combines the Ruby programming language with HTML, CSS, and JavaScript to create a web application that runs on a web server. When Rails is plugged into Ruby, it is often referred to as “Ruby on Rails”.
DEVELOPMENT AND VERSION CONTROL ENVIRONMENT	Git	Git is a version control system that is used for software development and other version control tasks. As a distributed revision control system. Git is free software distributed under the terms of the GNU General Public License version 2.
HOSTING SERVICE	GitHub	GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. GitHub offers both plans for private repositories and free accounts, which are usually used to host open-source software projects
DATABASE	PostgreSQL	PostgreSQL is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance. As a database server, its primary function is to store data securely, and to allow for retrieval at the request of other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.
WEB SERVER	NGINX	NGINX is a free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. NGINX is known for its high

		performance, stability, rich feature set, simple configuration, and low resource consumption
WEB SERVER SOFTWARE	Apache	Apache HTTP Server is the world's most used web server software. Apache is an open-source project. Runs on all major server operating systems.

Due to the variation in size between different deployments of 1-Click, load testing on specific configurations is required to give accurate assessments of hardware requirement. However, for comparison purposes, the 1-Click demonstration and quality assurance deployments are hosted at Heroku.com and Amazon Web Services with the following specifications:

Table 5: Server requirements

Device	Web Server	Storage	OTP Server	Database Server
Type	Heroku 2X Dyno	AWS S3	AWS m3.xlarge	Heroku 2X Dyno
CPU	8-core Intel Xeon E5-2680 v2 (Ivy Bridge)	2 virtual cores	13 virtual cores	8-core Intel Xeon E5-2680 v2 (Ivy Bridge)
Memory	2 GB	4 GB	15 GB	2 GB
Storage	100 GB+	100 GB	80 GB	512 GB
OS	Ubuntu v12.04 LTS (or later)	Ubuntu v12.04 LTS (or later)	Ubuntu v12.04 LTS (or later)	Ubuntu v12.04 LTS (or later)
Software	Apache	-	OpenTripPlanner	PostgreSQL 9.x (or later)

AWS Best Practice

1) Failover - Elastic IPs: Elastic IP is a static IP that is dynamically re-mappable. You can quickly remap and failover to another set of servers so that your traffic is routed to the new servers. It works great when you want to upgrade from old to new versions or in case of hardware failures.
2) Utilize multiple Availability Zones: Availability Zones are conceptually like logical datacenters. By deploying your architecture to multiple availability zones, you can ensure highly availability. Utilize Amazon RDS Multi-AZ] deployment functionality to automatically replicate database updates across multiple Availability Zones.
3) Maintain an Amazon Machine Image so that you can restore and clone environments very easily in a different Availability Zone; Maintain multiple Database slaves across Availability Zones and setup hot replication.
4) Utilize Amazon CloudWatch (or various real-time open source monitoring tools) to get more visibility and take appropriate actions in case of hardware failure or performance degradation. Setup an Auto scaling group to maintain a fixed fleet size so that it replaces unhealthy Amazon EC2 instances by new ones.
5) Utilize Amazon EBS and set up cron jobs so that incremental snapshots are automatically uploaded to Amazon S3 and data is persisted independent of your instances.

6) Utilize Amazon RDS and set the retention period for backups, so that it can perform automated backups.

3.2.1 Security Hardware Architecture

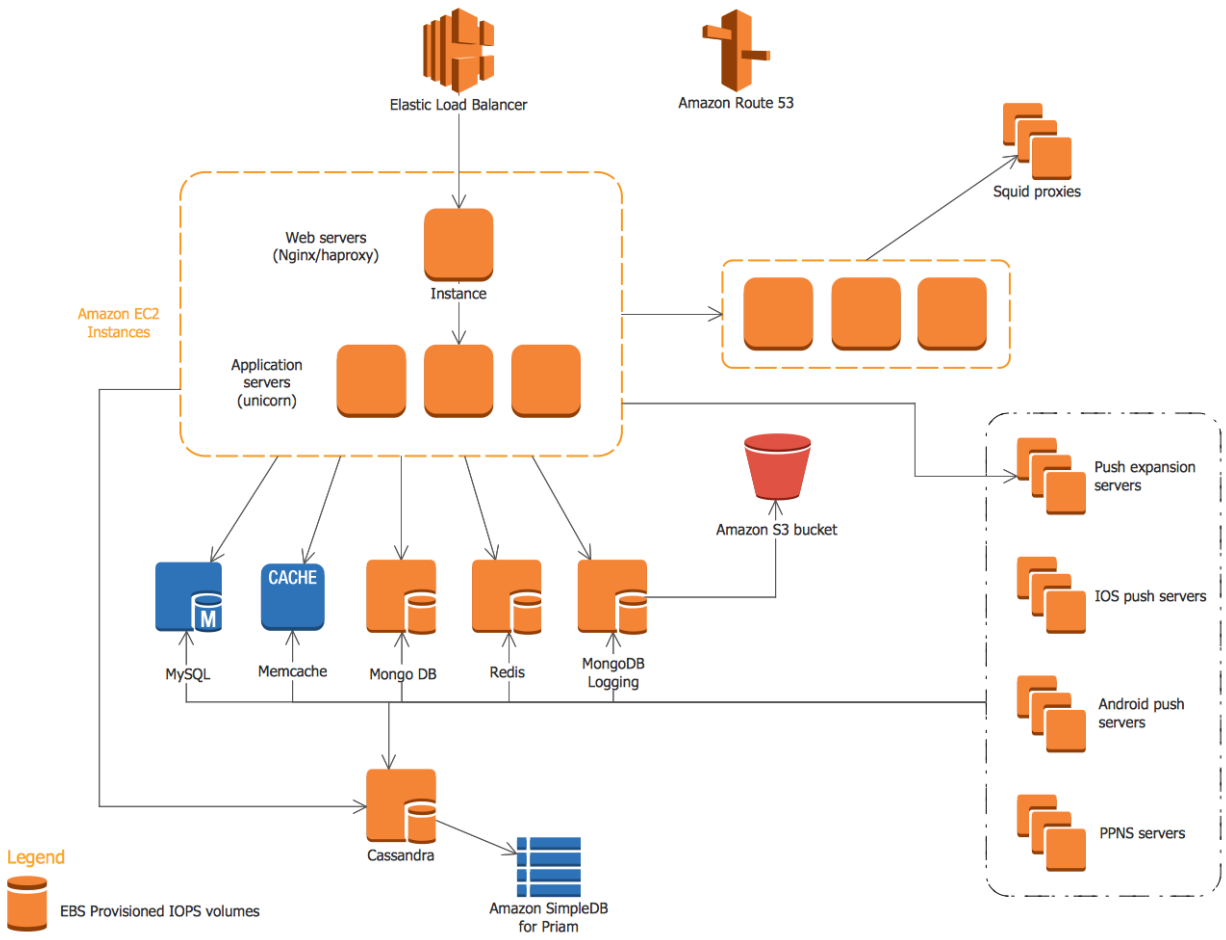


Figure 5: Security architecture

3.2.2 Performance Hardware Architecture

The current and proposed solution utilizes AWS S3 for hardware performance and reliability. Amazon S3 is storage for the Internet. It's a simple storage service that offers software developers a highly-scalable, reliable, and low-latency data storage infrastructure at very low costs.

Amazon S3 provides a simple web service interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. Using this web service, developers can easily build applications that make use of Internet storage. Since Amazon S3 is highly scalable and you only pay for what you use, developers can start small and grow their application as they wish, with no compromise on performance or reliability.

Amazon S3 is also designed to be highly flexible. Store any type and amount of data that you want; read the same piece of data a million times or only for emergency disaster recovery; build a simple FTP application, or a sophisticated web application such as the Amazon.com retail web site. Amazon S3 frees developers to focus on innovation, not figuring out how to store their data.

Amazon S3 gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. S3 Standard is designed for 99.99% availability and Standard - IA is designed for 99.9% availability. Both are backed by the Amazon S3 Service Level Agreement.

3.3 Software Architecture

The three-tier architecture is a popular pattern for user-facing applications. The tiers that comprise this architecture include the presentation tier, the logic tier, and the data tier. The presentation tier represents the component that users directly interact with (such as a web page, mobile app UI, etc.). The logic tier contains the code required to translate user actions at the presentation tier to the functionality that drives the application's behavior. The data tier consists of storage media (databases, object stores, caches, file systems, etc.) that hold the data relevant to the application.

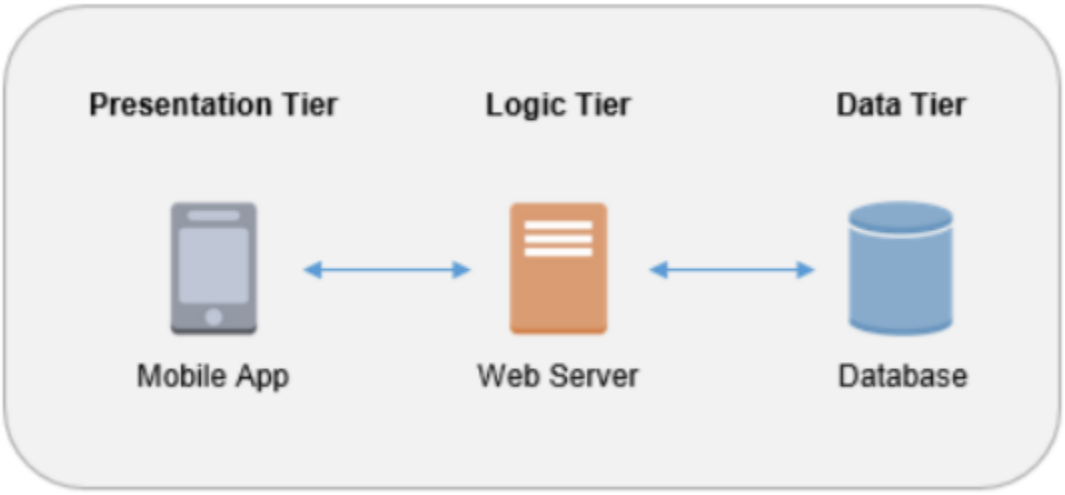


Figure 6: Software architecture tiers

The Serverless Logic Tier

The logic tier of the three-tier architecture represents the brains of the architecture. The features of the two services allow you to build a serverless production application that is highly available, scalable, and secure.

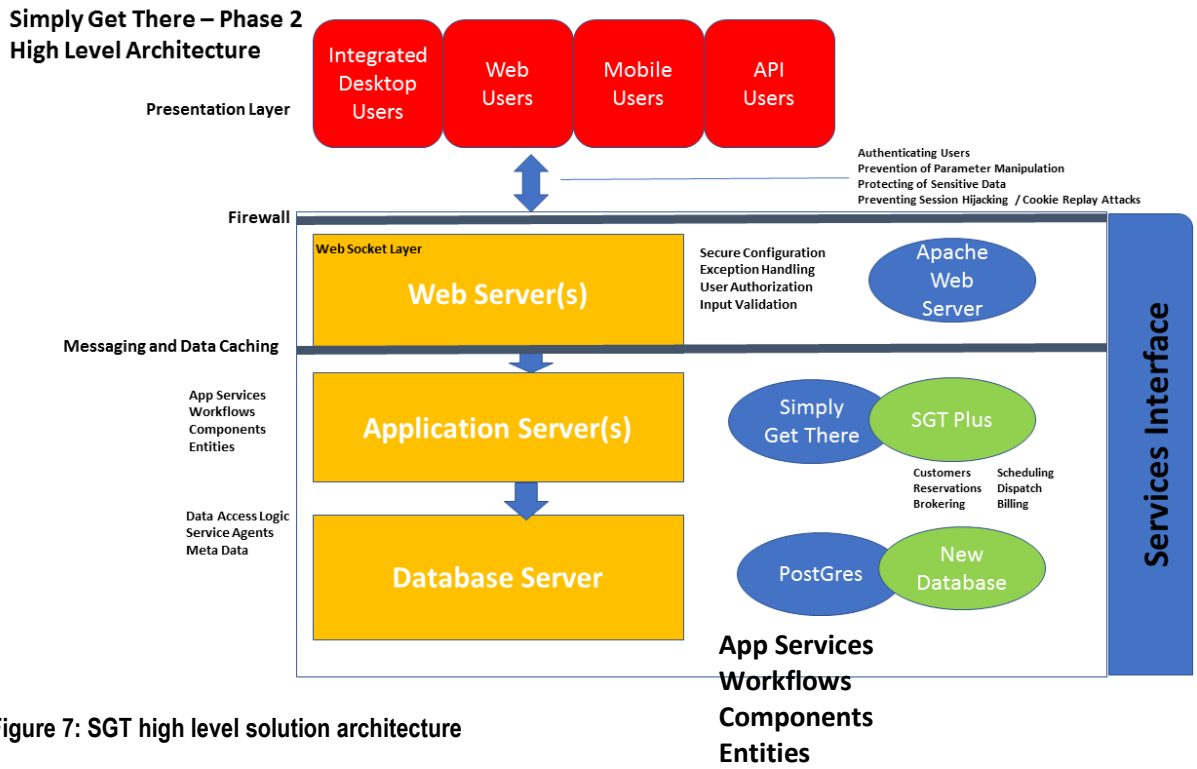


Figure 7: SGT high level solution architecture

3.3.1 Software Elements

Table 6: Software elements and descriptions

FUNCTION	DESCRIPTION
GENERAL ACCESS	User can access generally available website using any browser. www.simplygetthere.org
TRIP PLANNING / DISCOVERY	User can select multiple options to plan or discover a trip. Trip options include: • Bike • Drive • Specialized Services • Vehicle for Hire • Public Transit
ELIGIBILITY	If user selects the specialized service options, an eligibility form is displayed.
TRIP PLAN REVIEW	Based on user inputs a single or multiple trip plans will be displayed. Trip segments are listed by BUS, SPECIALIZED SERVICES, SUBWAY, WAIT, and WALK. Trip Options filters are provided.

TRIP DETAILS	Based on trip plan review and selected trip plan, the system will display the trip plan detail. If specialized transportation plan was selected, the system will display the origin and destination and the selected specialized transportation provider.
TRIP PLAN PRINT	User can choose to print the trip plan
TRIP PLAN EMAIL	User can choose to email the trip plan
TRAVEL PROFILE	If registered, user can maintain and manage their user profile.
TRIP PROFILE	User can view selected trip plans. Users can delete edit or remove the trip from the profile. User can get details of the planned trip.
PLACES	User can save common origins or destinations in the Places function to customize the planning process to their common travel plans.
PROVIDERS	Users can obtain a list of all transportation providers in the system with hyperlink to provider detailed information.

3.3.2 Security Software Architecture

There are a number of principles applied to current and proposed system security.

- Apply security at all layers:
    - Rather than running security appliances (e.g., firewalls) only at the edge of your infrastructure, use firewalls and other security controls on all of your resources (e.g., every virtual server, load balancer, and network subnet).
- Enable traceability:
    - Log and audit all actions and changes to your environment.
- Implement a principle of least privilege:
    - Ensure that authorization is appropriate for each interaction with your AWS resources and implement strong logical access controls directly on resources.

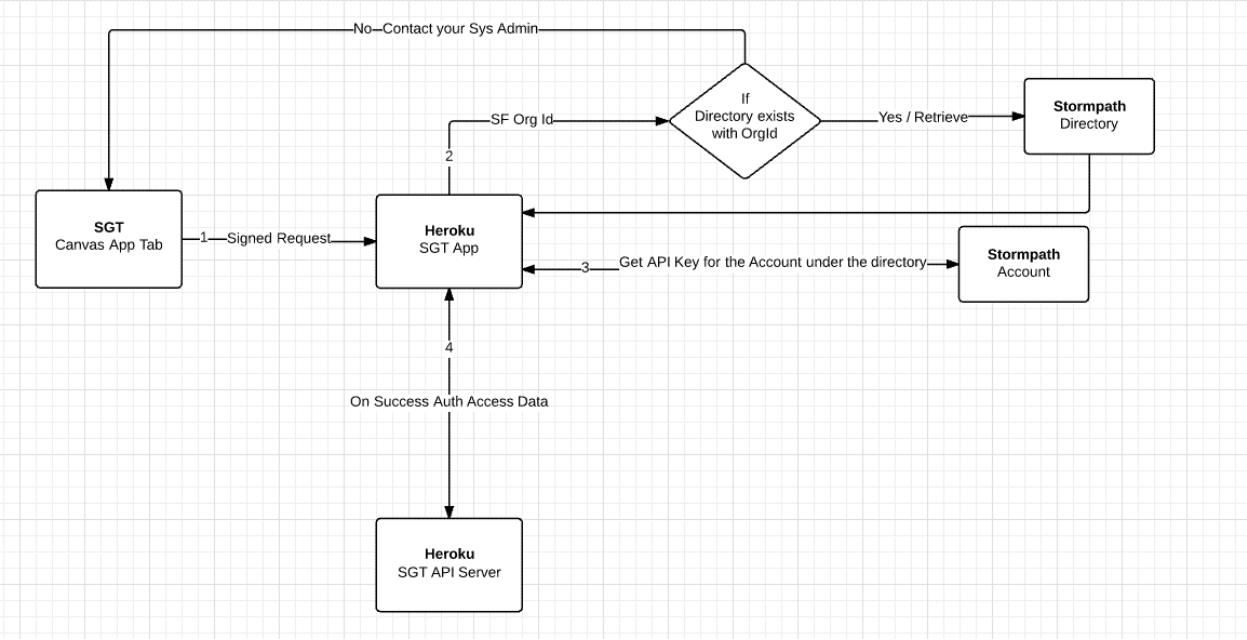


Figure 8: Security authentication

3.3.3 Performance Software Architecture

The most fundamental reason for performance concerns is that the tasks we set our systems to perform have become much more complex over time. The performance of the system depends on much more than the raw processing power of its hardware. The way that hardware is configured, the way resources are allocated and managed, and the way the software is written can have significant impacts on the system's ability to meet its performance goals.

The scalability property of a system is closely related to performance, but rather than considering how quickly the system performs its current workload, scalability focuses on the predictability of the system's performance as the workload increases.

Desired Quality	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes in the future if required
Applicability	Any system with complex, unclear, or ambitious performance requirements; systems whose architecture includes elements whose performance is unknown; and systems where future expansion is likely to be significant
Concerns	▪ response time ▪ throughput ▪ scalability ▪ predictability ▪ hardware resource requirements ▪ peak load behavior
Activities	▪ capture the performance requirements ▪ create the performance models ▪ analyze the performance models ▪ conduct practical testing ▪ assess against the requirements ▪ rework the architecture
Tactics	▪ optimize repeated processing ▪ reduce contention via replication ▪ prioritize processing ▪ consolidate related workload ▪ distribute processing over time ▪ minimize the use of shared resources ▪ reuse resources and results ▪ partition and parallelize ▪ scale up or scale out ▪ degrade gracefully ▪ use asynchronous processing ▪ relax transactional consistency ▪ make design compromises
Pitfalls	▪ imprecise performance and scalability goals ▪ unrealistic models ▪ use of simple measures for complex cases ▪ inappropriate partitioning ▪ invalid environment and platform assumptions ▪ too much indirection ▪ concurrency-related contention ▪ database contention ▪ transaction overhead ▪ careless allocation of resources ▪ disregard for network and in-process invocation differences

Figure 9: Performance architecture scalability

3.4 Information Architecture

There are minimal additional personal data elements that will be stored in the new functionality. SGT currently stores customer and transportation related data of the consumer. This data is classified as personally identifiable information. Minor health related records are stored with this information. Health information can be derived based on location information.

The proposed new features and functions of the system will include additional PII data for the trip reservation function. This data includes:

- Trip Purpose

- Trip Type

3.4.1 Records Management

HIPAA is a major federal records management regulation that must be considered and adhered to

3.4.1.1 Data

Data is supplied by end user or the consumer of the system. Data is entered into the system via the web application. Call center users will also enter data into the system via a graphical user interface. Third party provider data may also be inserted into the data via API's. API middleware for trip and coordination transactions will manage the exporting and importing of any third party data elements.

3.4.1.2 Manual/Electronic Inputs

All inserts or upserts into database shall be managed using industry standard data validation tools and triggers. Data validation is intended to provide certain well-defined guarantees for fitness, accuracy, and consistency for any of various kinds of user input into an application or automated system. Data validation rules can be defined and designed using any of various methodologies, and be deployed in any of various contexts.

Types:

- Data type validation;
- Range and constraint validation;
- Code and Cross-reference validation; and
- Structured validation

Data-type validation

Data type validation is customarily carried out on one or more simple data fields. The simplest kind of data type validation verifies that the individual characters provided through user input are consistent with the expected characters of one or more known primitive data types; as defined in a programming language or data storage and retrieval mechanism. As an example, telephone numbers are routinely expected to include the digits and possibly the characters +, -, ( ) (plus, minus, and parentheses).

Simple range and constraint validation

Simple range and constraint validation may examine user input for consistency with a minimum/maximum range, or consistency with a test for evaluating a sequence of characters, such as one or more tests against regular expressions.

Code and cross-reference validation

Code and cross-reference validation includes tests for data type validation, combined with one or more operations to verify that the user-supplied data is consistent with one or more external rules, requirements, or validity constraints relevant to a particular organization, context or set of underlying assumptions. These additional validity constraints may involve cross-referencing supplied data with a known look-up table or directory information service such as LDAP.

Structured validation

Structured validation allows for the combination of any of various basic data type validation steps, along with more complex processing. Such complex processing may include the testing of conditional constraints for an entire complex data object or set of process operations within a system.

3.4.1.3 Master Files

The following tables define the data maintained in the proposed system.

3.5 Internal Communications Architecture

Current and proposed solution is managed in AWS. Specific network architecture is not provided due to security issues.

3.6 Security Architecture

Not available

3.7 Performance

Not available

4. System Design

The proposed system may extend the current system via the existing SGT framework or by the development of a modular component that “plugs” into the SGT application. Proposed solution may be commercially available or custom developed to fully meet the requirements of the project. ARC will host the application internally but may also choose to host the solution in a third party environment. Technical support and maintenance will be required for the transactional and mission critical components of the system. The system must provide strong security and credentialing methods to ensure privacy and system security.

4.1 Business Requirements

System could provide substantial functionality to the existing SGT application. ARC intends to understand the functional requirements necessary for each component and its applicable capital and ongoing support costs to layout a short to mid-range implementation plan.

Table 7: SGT future enhancements

WEB BASED RESERVATIONS	**1**
PROVIDER ASSIGNMENT	2
RESOURCE DATA MAINTENANCE	3
TRIP DATA EXCHANGE	4
CENTRALIZED ELIGIBILITY	5
TNC MODE INTEGRATION	6
REPORTING AND ANALYTICS	7
AUTOMATED SCHEDULING	8
ROUTE PLANNING AND OPTIMIZATION	9
AUTOMATED DISPATCHING	10
AUTOMATED VEHICLE TRACKING	11

MOBILE DRIVER APP	12
CUSTOMER INFORMATION APP	13

4.2 Database Design

Data dictionary is provided as an attachment to this document.

Figure 10: Example of a database design

4.2.1 Data Objects and Resultant Data Structures

The following table defines the data objects and schema for the proposed solution.

Table 8: Data objects and schemas

```
ActiveRecord::Schema.define(version: 20170419145226) do

      xtensions that must be enabled in order to support this database
```

```
ion "plpgsql"
ion "postgis"


"accommodations", force: :cascade do |t|
"code",       null: false
"created_at", null: false
"updated_at", null: false
ode"], name: "index_accommodations_on_code", unique: true, using: :btree



"accommodations_services", id: false, force: :cascade do |t|
service_id",       null: false
accommodation_id", null: false
ccommodation_id"], name: "index_accommodations_services_on_accommodation_id", using:
  :btree
ervice_id"], name: "index_accommodations_services_on_service_id", using: :btree



"accommodations_users", id: false, force: :cascade do |t|
user_id",          null: false
accommodation_id", null: false
ccommodation_id"], name: "index_accommodations_users_on_accommodation_id", using:
  :btree
ser_id"], name: "index_accommodations_users_on_user_id", using: :btree



"cities", force: :cascade do |t|
"name"
"state"
"geom",       limit: {:srid=>0, :type=>"geometry"}
"created_at",                                    null: false
"updated_at",                                    null: false
eom"], name: "index_cities_on_geom", using: :gist
ame", "state"], name: "index_cities_on_name_and_state", using: :btree



"comments", force: :cascade do |t|
"comment"
```

```
"locale"
"commentable_type"
"commentable_id"
"created_at",       null: false
"updated_at",       null: false
ommentable_type", "commentable_id"], name:
  "index_comments_on_commentable_type_and_commentable_id", using: :btree



"configs", force: :cascade do |t|
"created_at", null: false
"updated_at", null: false
"key"
"value"



"counties", force: :cascade do |t|
"name"
"state"
"geom",       limit: {:srid=>0, :type=>"geometry"}
"created_at",                                       null: false
"updated_at",                                       null: false
eom"], name: "index_counties_on_geom", using: :gist
ame", "state"], name: "index_counties_on_name_and_state", using: :btree



"custom_geographies", force: :cascade do |t|
"name"
"geom",       limit: {:srid=>0, :type=>"geometry"}
"created_at",                                       null: false
"updated_at",                                       null: false
eom"], name: "index_custom_geographies_on_geom", using: :gist
ame"], name: "index_custom_geographies_on_name", using: :btree



"eligibilities", force: :cascade do |t|
"code",       null: false
"created_at", null: false
"updated_at", null: false
```

```
ode"], name: "index_eligibilities_on_code", unique: true, using: :btree



"eligibilities_services", id: false, force: :cascade do |t|
service_id",     null: false
eligibility_id", null: false
ligibility_id"], name: "index_eligibilities_services_on_eligibility_id", using:
  :btree
ervice_id"], name: "index_eligibilities_services_on_service_id", using: :btree



"fare_zones", force: :cascade do |t|
"service_id"
"region_id"
"code"
"created_at", null: false
"updated_at", null: false
ervice_id", "region_id"], name: "index_fare_zones_on_service_id_and_region_id",
  using: :btree



"itineraries", force: :cascade do |t|
"created_at",   null: false
"updated_at",   null: false
"trip_id"
"start_time"
"end_time"
"legs"
"walk_time"
"transit_time"
"cost"
"service_id"
"trip_type"
ervice_id"], name: "index_itineraries_on_service_id", using: :btree
rip_id"], name: "index_itineraries_on_trip_id", using: :btree



"landmarks", force: :cascade do |t|
"created_at",                          null: false
```

```
"updated_at",                              null: false
"name"
"street_number"
"route"
"city"
"state"
"zip"
"old"
"lat",           precision: 10, scale: 6
"lng",           precision: 10, scale: 6



"locales", force: :cascade do |t|
"name"
"created_at", null: false
"updated_at", null: false



"purposes", force: :cascade do |t|
"code",       null: false
"created_at", null: false
"updated_at", null: false



"purposes_services", id: false, force: :cascade do |t|
service_id", null: false
purpose_id", null: false
urpose_id"], name: "index_purposes_services_on_purpose_id", using: :btree
ervice_id"], name: "index_purposes_services_on_service_id", using: :btree



"regions", force: :cascade do |t|
"recipe"
"geom",       limit: {:srid=>0, :type=>"multi_polygon"}
"created_at",                                          null: false
"updated_at",                                          null: false
eom"], name: "index_regions_on_geom", using: :gist
```

```
"roles", force: :cascade do |t|
"name"
"resource_type"
"resource_id"
"created_at"
"updated_at"
ame", "resource_type", "resource_id"], name:
  "index_roles_on_name_and_resource_type_and_resource_id", using: :btree
ame"], name: "index_roles_on_name", using: :btree



"schedules", force: :cascade do |t|
"service_id"
"day"
"start_time"
"end_time"
"created_at", null: false
"updated_at", null: false
ay"], name: "index_schedules_on_day", using: :btree
ervice_id"], name: "index_schedules_on_service_id", using: :btree



"services", force: :cascade do |t|
"created_at",                          null: false
"updated_at",                          null: false
"type"
"name"
"gtfs_agency_id"
"logo"
"email"
"url"
"phone"
"start_or_end_area_id"
"trip_within_area_id"
"fare_structure"
"fare_details"
"archived",             default: false
rchived"], name: "index_services_on_archived", using: :btree
tfs_agency_id"], name: "index_services_on_gtfs_agency_id", using: :btree
ame"], name: "index_services_on_name", using: :btree
```

```
tart_or_end_area_id"], name: "index_services_on_start_or_end_area_id", using: :btree
rip_within_area_id"], name: "index_services_on_trip_within_area_id", using: :btree



"translation_keys", force: :cascade do |t|
"name"
"created_at", null: false
"updated_at", null: false



"translations", force: :cascade do |t|
"locale_id"
"translation_key_id"
"value"
"created_at",         null: false
"updated_at",         null: false



"trips", force: :cascade do |t|
"created_at",            null: false
"updated_at",            null: false
"user_id"
"origin_id"
"destination_id"
"trip_time"
"arrive_by"
"selected_itinerary_id"
"purpose_id"
estination_id"], name: "index_trips_on_destination_id", using: :btree
rigin_id"], name: "index_trips_on_origin_id", using: :btree
urpose_id"], name: "index_trips_on_purpose_id", using: :btree
elected_itinerary_id"], name: "index_trips_on_selected_itinerary_id", using: :btree
ser_id"], name: "index_trips_on_user_id", using: :btree



"user_eligibilities", force: :cascade do |t|
"user_id"
"eligibility_id"
"value",          default: true
```

```
"created_at",                    null: false
"updated_at",                    null: false
ligibility_id"], name: "index_user_eligibilities_on_eligibility_id", using: :btree
ser_id"], name: "index_user_eligibilities_on_user_id", using: :btree



"users", force: :cascade do |t|
"created_at",                                  null: false
"updated_at",                                  null: false
"email",                           default: "", null: false
"encrypted_password",              default: "", null: false
"reset_password_token"
"reset_password_sent_at"
"remember_created_at"
"sign_in_count",                   default: 0,  null: false
"current_sign_in_at"
"last_sign_in_at"
"current_sign_in_ip"
"last_sign_in_ip"
"authentication_token",   limit: 30
"first_name"
"last_name"
"preferred_locale_id"
"preferred_trip_types"
uthentication_token"], name: "index_users_on_authentication_token", unique: true,
  using: :btree
mail"], name: "index_users_on_email", unique: true, using: :btree
ast_name", "first_name"], name: "index_users_on_last_name_and_first_name", using:
  :btree
referred_locale_id"], name: "index_users_on_preferred_locale_id", using: :btree
eset_password_token"], name: "index_users_on_reset_password_token", unique: true,
  using: :btree



"users_roles", id: false, force: :cascade do |t|
user_id"
role_id"
ser_id", "role_id"], name: "index_users_roles_on_user_id_and_role_id", using: :btree
```

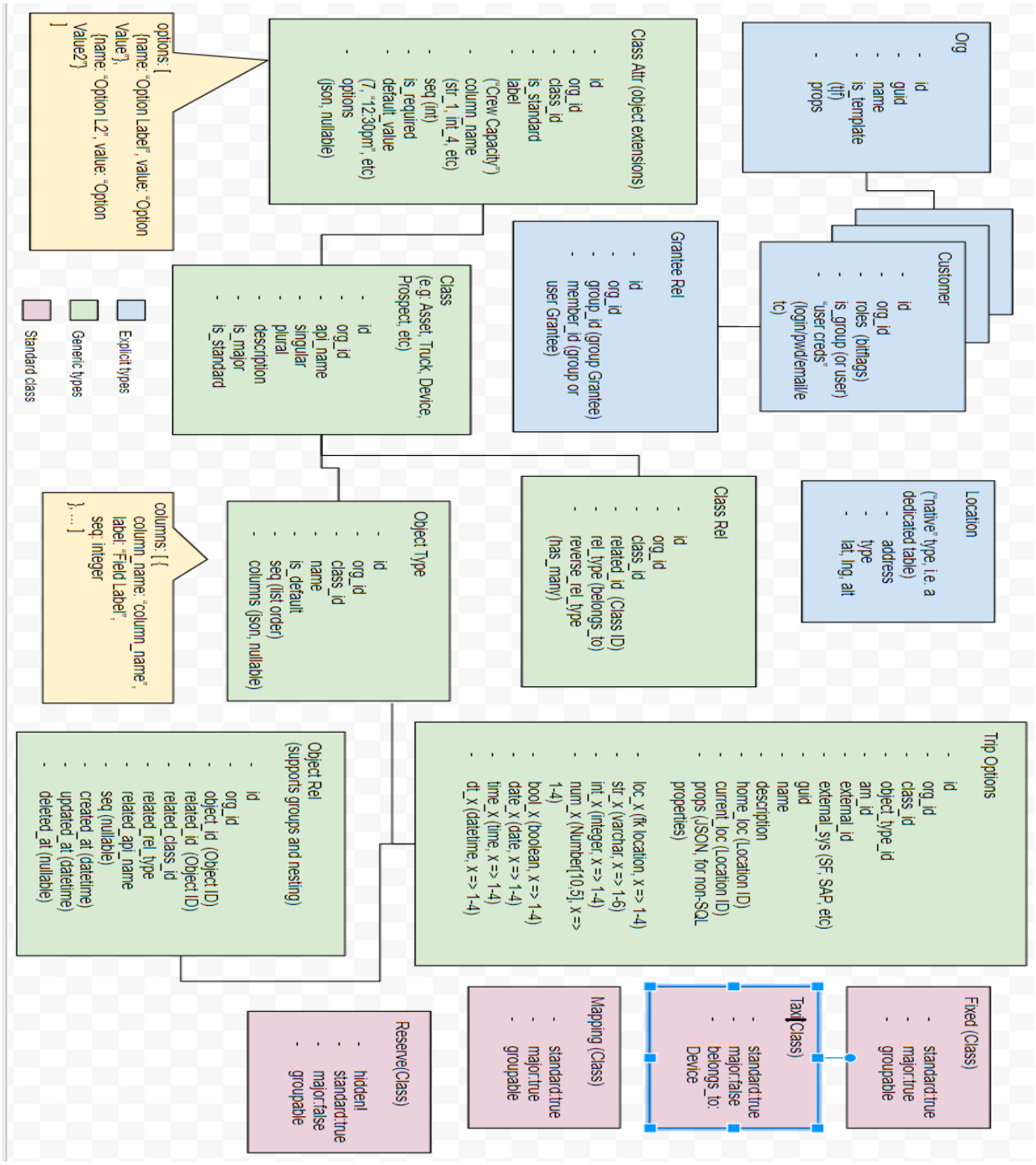
```
"waypoints", force: :cascade do |t|
"created_at",                              null: false
"updated_at",                              null: false
"name"
"street_number"
"route"
"city"
"state"
"zip"
"lat",           precision: 10, scale: 6
"lng",           precision: 10, scale: 6



"zipcodes", force: :cascade do |t|
"name"
"geom",       limit: {:srid=>0, :type=>"geometry"}
"created_at",                                        null: false
"updated_at",                                        null: false
eom"], name: "index_zipcodes_on_geom", using: :gist
ame"], name: "index_zipcodes_on_name", using: :btree



ey "itineraries", "services"
ey "itineraries", "trips"
ey "schedules", "services"
ey "services", "regions", column: "start_or_end_area_id"
ey "services", "regions", column: "trip_within_area_id"
ey "trips", "itineraries", column: "selected_itinerary_id"
ey "trips", "purposes"
ey "trips", "users"
ey "trips", "waypoints", column: "destination_id"
ey "trips", "waypoints", column: "origin_id"
ey "user_eligibilities", "eligibilities"
ey "user_eligibilities", "users"
ey "users", "locales", column: "preferred_locale_id"
```

4.2.2 File and Database Structures

The following data model represents the current solution with the proposed solution highlighted.

4.2.2.1 Database Management System Files

The image below represents the high level ER diagram and object model for the current system.



4.2.2.2 Non-Database Management System Files

None

4.3 Data Conversion

None

4.4 User Machine-Readable Interface

4.4.1 Inputs

See Concept of Operations for user inputs, forms, and user roles and permissions.

4.4.2 Outputs

Instructions: Describe the system output design relative to the user/operator. Show a mapping to the high-level data flows. System outputs include reports, data display screens and GUIs, query results, etc. The output files described in the section for Data Design may be referenced. The following should be provided, if appropriate:

- *Identification of codes and names for reports and data display screens*
- *Description of report and screen contents (provide a graphical representation of each layout and define all data elements associated with the layout or reference the data dictionary)*
- *Description of the purpose of the output, including identification of the primary users*
- *Report distribution requirements, if any (include frequency for periodic reports)*
- *Description of any access restrictions or security considerations*

Users did not specifiy specific outputs. They requested that an ad hoc report and dashboard design tool . This would allow them to define and build their own reports and queries. The current application provides standard reports that are unusable and unchangeable. This is a big area of frustration.

The report designer will expose all available objects and fields including extensions and allow admin users to sort, group, filter, and bucket datasets. Basic and advanced statistics and data aggregration functions will be available. Reports can be defined as tabular, summary, matrix, and joined. The report tool will allow a basic user to build simple and/or very advanced reports. This method provides the flexibility and scalability ARC requires.

4.5 User Interface Design

User interface designs have not been completed for this document. It is anticipated that the development team will develop mockups based on final procurement specifications. The image below represents and example mockup.

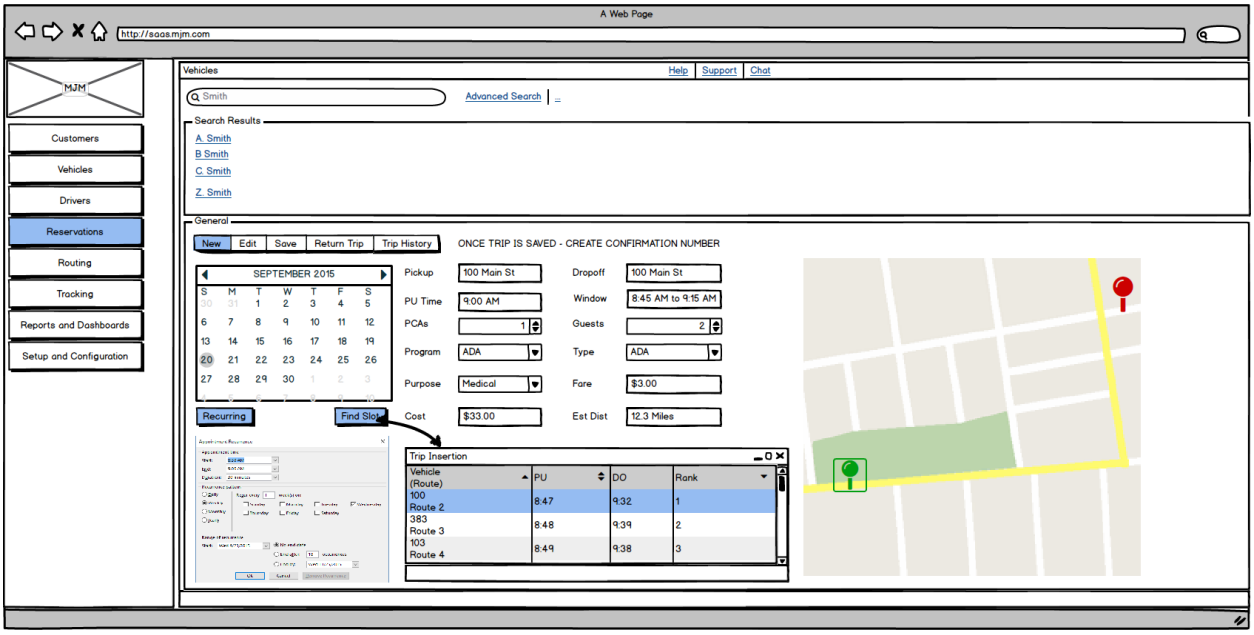


Figure 11: Example of balsamiq mockup

4.5.1 Section 508 Compliance

The current application is Section 508 Compliant. The proposed application will be designed and developed to support Section 508 compliance.

5. Operational Scenarios

Section 5 describes scenarios that show how the system will be used to perform its objectives and meet the user's requirements. Each scenario can be illustrated by a use case. Develop sample usage scenarios (as realistic as possible) for each user class that show what inputs, outputs, and user interaction will be required.

1.1 Major Operational Scenarios

Table 9: Major operational scenarios

Process	Purpose	Description	Priority	Frequency
Reservations	Web based trip booking	Users can book transportation directly from trip planning function. Users may choose to book directly without going through planning process.	High	Daily
Centralized Resource Management	Central repository for customer, vehicle, and provider data	System can leverage regional resources more effectively. Foundation for regional coordination and provider assignment.	High	Daily
Provider Assignment	Trip Assignment and Brokering	Functions to support automated trip assignment based on least cost most appropriate logic.	High	Daily
Scheduling	Schedule and route planning	Functions to support automated, computer assisted, and manual scheduling and route optimization to coordinate trips and improve capacity.	High	Daily
Dispatching	Real time dispatching and tracking	Users can view real time location, status, and ETA of transportation to improve service delivery and customer service.	High	Daily
Electronic Payment	Web or mobile payment	Users can pay for transportation online via credit card or, potentially, a pre-paid transportation account.	High	Daily
Cost Allocation	Transportation cost sharing	System can allocate costs to proper funding source.	High	Daily
Reporting	Reporting and Analytics	Users can run canned reports and dashboards. Users can create custom ad hoc reports.	High	Daily / Weekly / Monthly

Coordination	Trip Coordination	Trip data can be exchanged electronically via published and open API's for the facilitation of coordination of resources and trips.	High	Daily

1.2 Major Use Cases

1.2.1 Customer Resources

Creating and editing customer demographics including default address information (geocoded), eligibility information, capacity and constraint related parameters (mobility needs, PCA's, guests, etc.…), trip related data (trip purpose, trip type), and billing information. Information below defines the core data management requirements for customers.

Major Functions:

- SEARCH - Searching for Customers
- NEW - Creating New Customers
- EDIT - Editing Customers
- DELETE - Deleting Customers

Table 10: Customer use cases

• Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the customer module	I can add, edit, or review customer related information	**<given>**a user with rights to customer module needs to launch customer module**<when>**the user selects a customer tab/button/section**<then>**customer module is displayed	UI design for module accessibility will be important
User Interface	Search for customer	Edit an existing customer	**<given>**a user needs to quickly search for a single or multiple customers **<when>**the user provides full or partial name **<then>**the application provides a single or multiple customers for user to select	Search UI must be simple and fast. Wildcard searches required. Advanced search criteria required
User Interface	Create New Customer	I can add a new customer to the database	**<given>**a user has rights to create new customer**<when>**the user selects NEW function**<then>**the application provides new form to enter required data	

General Data	Edit or enter basic customer demographic data	I can store customer data for future use	**<given>** user has rights to create and edit customer data **<when>**the user creates new or edits a customer **<then>** the application allows user to input data in form: First Name, Last Name, Default Pickup Address, Mailing Address, Phone, Email, and Birthday	
User Interface	Delete customer	I can purge customer from system	**<given>** user has rights to delete customer **<when>**the user creates selects a customer **<then>** the application allows user to delete customer	All associated data, including trips, must be purged. Warning message should be displayed prior to submitting request
General Data	Add Language to customer record	I can assign default language to customer	**<given>** user has rights to create and edit customer data **<when>**the user creates new or edits a customer **<then>** allow user to select default LANGUAGE from picklist values	
General Data	Add RACE to customer record	I can assign default race to customer	**<given>** user has rights to create and edit customer data **<when>**the user creates new or edits a customer <**then>** allow user to select default RACE from picklist values	
General Data	Add GENDER to customer record	I can assign default gender to customer	**<given>** user has rights to create and edit customer data **<when>**the user creates new or edits a customer <**then>** allow user to select default GENDER from picklist values	
General Data	Add emergency contact	I can assign a default emergency contact to customer	**<given>** user has rights to create and edit customer data **<when>**the user creates new or edits a customer <**then>** allow user to assign an emergency contact with phone number to record	

Function	Geocode a customer address (address, city, state, zip)	I can locate default customer pickup address on map	**<given>**user has rights to create new customer <**when>** user enters a valid pickup address **<then>** locate the customer on the map and provide user with options to select appropriate address	Design consideration: Address geocoding must be very simple and straightforward. Reducing address redundancy is important (i.e. Address duplication)
Function	Calculate customer age	I can easily determine the age of the customer as a function of AGE field	**<given>**user has rights to create or edit customers **<when>**user enters AGE data <**then>** then application calculates age of customer	Used in reporting especially in senior and social services transportation. Often used to determine eligible programs
Function	Make customer active	I can easily make a customer active or inactive	**<given>**the user has rights to create or edit customer <**when>**user creates new customer **<then>** automatically make customer active	Upon NEW make customer active. Allow user to change to inactive in future. If changed to INACTIVE, all future trip reservations should be cancelled
General Data	Add a customer picture to customer record	I can view image of customer	**<given>**the user has rights to create or edit customer **<when>**the user creates or edits customer**<then>** provide method to upload customer image to record	Customer image may also be utilized for future mobile data app to identify customer upon boarding
Mobility	Add customer mobility type	I can assign mobility type to customer	**<given>**the user has rights to create or edit customer **<when>**the user creates or edits a customer **<then>** allow user to select MOBILITY TYPE from picklist values	Default picklist values: Ambulatory, Wheelchair, Extra Large Wheelchair

Mobility	Add load and unload type as a function of mobility type	I can assign automatically assign load and unload time to a customer profile for use in reservation and scheduling module	**<given>**the user has rights to create or edit customer **<when>**the user selects MOBILITY Type**<then>** assign default load and unload times as function of MOBILITY TYPE as default, but editable, values	Mobility type and load and unload time will populate automatically in RESERVATION module upon creating NEW RESERVATION
Mobility	Add additional passengers	I can add a Personal Care Attendant (PCA), Guest, or other passengers travelling with me	**<given>**the user has rights to create or edit customer **<when>**the user selects ADD PASSENGERS**<then>** allow user to enter additional passenger name, type (PCA, Guest)	Vehicle Capacity Constraint
Mobility	Add additional passenger's mobility type	I can add MOBILITY TYPES to any additional passenger travelling with me as a default	**<given>**the user has rights to create or edit customer **<when>**the user selects ADD PASSENGERS **<then>** allow user to enter mobility types for each passenger.	Vehicle Capacity Constraint
General Data	Add a customer type to customer record	I can add a customer type from a PICKLIST	**<given>**the user has rights to create or edit customer **<when>**the user selects TYPE picklist **<then>** allow user to enter a customer type from default picklist values	Placeholder to put customers in generic "buckets" - Elderly, Child, etc.…
General Data	Add assistance needs to customer record	I can add multiple assistance needs to customer record	**<given>**the user has rights to create or edit customer **<when>**the user selects ADD ASSISTANCE **<then>** allow user to add multiple ASSISTANCE NEED items to record	Assistance needs will be used for driver instructions and will be available on paper manifest or mobile device
General Data	Add COMMENTS to customer record	I can add general comments to customer record	**<given>**the user has rights to create or edit customer **<when>**the user selects COMMENTS field **<then>** allow user to enter text in COMMENTS field	

Function	Upload ATTACHMENTS to customer record	I can easily add or delete one or multiple documents to the customer record	**<given>**the user has rights to create or edit customer **<when>**the user **<then>** allow	
Eligibility	Add PROGRAM ELIGIBILITY to customer record	I can track which programs the customer is eligible for and to associate relevant eligibility information to future trip reservations	**<given>**the user has rights to add eligibility info to customers**<when>**the user selects ADD ELIGIBILITY **<then>** allow user to add multiple ELIGIBILITY records	
Eligibility	Add PROGRAM ELIGIBILITY details to ELIGIBILITY record	I can track various data elements relating to ELIGIBILITY include: Program (picklist), Start, End, Fare	**<given>**the user has rights to add eligibility info to customers**<when>**the user selects ELIGIBILITY record **<then>** allow user to enter relevant program eligibility data	Design consideration: Provide flexibility to allow admins to configure different eligibility criteria for each program. ADA has different criteria than Medicaid. Based on program selected, form is displayed based on configured layout / fields.
Function	Implement Audit Trail functions at field level	I can track and report on who created, updated, or deleted customer related data	**<given>**any user with Read Only rights**<when>**the user selects a customer record**<then>**display AUDIT TRAIL information	Audit trail should display both the user and the date/time of edit as well as data changed from -> to
User Interface	Implement inline and online help	I can obtain help at the form and field level	**<given>**any user with Read Only rights**<when>**the user selects HELP **<then>**display help at form or field level depending on what user requests	Online, Inline, and CBT will be very important in scaling and streamlining implementation

User Interface	Create or Cancel Trip Reservation	I can select a BUTTON to create or cancel a reservation directly from customer module	**<given>**the user has rights to create or edit customer **<when>**the user **<then>** allow	See RESERVATION Module.
User Interface	Add custom fields to the customer module	I can track data elements that are specific to my organizatio n	**<given>**the user is an admin **<when>**the user accesses ADMIN MODULE**<then>** allow user to define custom data elements to customer module	See ADMINISTRATIO N Module
User Interface	Configure picklist values	I can configure picklist values specific to my organizatio n	**<given>**the user has rights to create or edit customer **<when>**the user **<then>** allow	See ADMINISTRATIO N Module

1.2.2 Vehicle Resources

Vehicle module simply provides users the ability to define the type of vehicles operated and their relevant characteristics. This data is very important for the route and schedule optimization problem. Vehicle data will be passed into scheduling tools. Much of the data below is required by FTA for National Transit Database (NTD) annual reporting. Automated NTD reports will be critical and strong value add.

Key considerations:

- Vehicle Capacity
  - Ambulatory Seats
  - Wheelchair Slots
- Vehicle Availability
- Vehicle Requirements
  - Drivers must have this capability in order to be assigned to it
- Vehicle Pull In / Pullout (Garage Location)

Table 11: Vehicle use cases

Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the vehicle module	I can add, edit, or review vehicle	<given>a user with rights to vehicle module needs to launch vehicle module<when>the user selects a vehicle	UI design for module accessibility will be important

		related information	tab/button/section<then>vehicle module is displayed	
User Interface	Search for vehicle	Edit an existing vehicle	<given>a user needs to quickly search for a single or multiple vehicles <when>the user provides full or partial name <then>the application provides a single or multiple vehicles for user to select	Search UI must be simple and fast. Wildcard searches required. Advanced search criteria required
User Interface	Create New vehicle	I can add a new vehicle to the database	<given>a user has rights to create new vehicle<when>the user selects NEW function<then>the application provides new form to enter required data	
General Data	Edit or enter basic vehicle data	I can store vehicle data for future use	<given> user has rights to create and edit vehicle data <when>the user creates new or edits a vehicle <then> the application allows user to input data in form: Vehicle Number, Type, Fleet, VIN, Plate, Status, Make, Model, Fuel Type, Year, Color, Length, Ownership, Operating cost	Multiple Generic Vehicle fields
User Interface	Delete vehicle	I can purge vehicle from system	<given> user has rights to delete vehicle <when>the user creates selects a vehicle <then> the application allows user to delete vehicle	All associated data, including trips, must be unscheduled. Warning message should be displayed prior to submitting request
Function	Geocode a vehicle garage (address, city, state, zip)	I can locate default garage address on map	<given>user has rights to create new vehicle <when> user enters a valid pickup address <then> locate the vehicle on the map and provide user with options to select appropriate address	Design consideration: Address geocoding must be very simple and straightforward. Reducing address redundancy is important (i.e. Address duplication)
Function	Make vehicle active	I can easily make a vehicle active or inactive	<given>the user has rights to create or edit vehicle <when>user creates new vehicle <then> automatically make vehicle active	Upon NEW make vehicle active. Allow user to change to inactive in future. If changed to INACTIVE, all future trip reservations should be cancelled

General Data	Add a vehicle picture to vehicle record	I can view image of vehicle	<given>the user has rights to create or edit vehicle <when>the user creates or edits vehicle<then> provide method to upload vehicle image to record	vehicle image may also be utilized for future mobile data app to identify vehicle upon boarding
General Data	Add vehicle capacity	I can enter the number of ambulatory and wheelchair seats the vehicle has	<given>the user has rights to edit vehicle <when>the user creates or edits vehicle<then> allow user to enter capacity information - 1) AMB 2) WC	
General Data	Add equipment types the vehicle supports	I can assign multiple types of equipment to the vehicle	<given> user has rights to edit vehicle data <when>the user creates new or edits a vehicle <then> allow user to select one or multiple equipment types	Multi Picklist. o Constraints used in optimization to ensure vehicle has proper equipment to perform trips assigned. For example, a wheelchair trip can only be assigned to a vehicle that has both a wheel chair lift (equipment) and a slot / seat (capacity) at that given time.
General Data	Add vehicle restrictions	I can assign restrictions to the vehicle. This tells the system what the vehicle is not allowed to do.	<given> user has rights to edit vehicle data <when>the user selects vehicle and vehicle schedule tab <then> allow user to select one or multiple vehicle restrictions	Multi Picklist. Values could equal - No U-Turns, Toll Roads, Out of State Trips, etc.…
General Data	Assign vehicle to a route or run	I can assign a vehicle to a route for a single day or multiple days	<given> user has rights to edit vehicle data <when>the user selects vehicle and vehicle schedule tab <then>allow user to assign the vehicle to a defined route	NOTE: Need to define a route object
General Data	Assign a pull out cost to vehicle	I can assign a "cost" that weights the vehicles pull out assignment	<given> user has rights to edit vehicle data <when>the user selects vehicle and vehicle schedule tab <then>allow user to assign a numeric cost to the vehicle	Cost will be used to determine least costly vehicles to utilize when assigning schedules

1.2.3 Driver Resource

Driver module allows users to maintain a list of drivers and related information.

COMMON ACTIONS

- NEW
- SEARCH
    - Easy and flexible search functions.
    - Once driver is identified, user can edit record
        - EDIT
        - DELETE
        - VIEW SCHEDULE

Table 12: Driver resource use cases

Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the driver module	I can add, edit, or review driver related information	<given>a user with rights to driver module needs to launch driver module<when>the user selects a driver tab/button/section<then>driver module is displayed	UI design for module accessibility will be important
User Interface	Search for driver	Edit an existing driver	<given>a user needs to quickly search for a single or multiple drivers <when>the user provides full or partial name <then>the application provides a single or multiple drivers for user to select	Search UI must be simple and fast. Wildcard searches required. Advanced search criteria required
User Interface	Create New driver	I can add a new driver to the database	<given>a user has rights to create new driver<when>the user selects NEW function<then>the application provides new form to enter required data	
General Data	Edit or enter basic driver data	I can store driver data for future use	<given> user has rights to create and edit driver data <when>the user creates new or edits a driver <then> the application allows user to input data in form: Name, Address, Phone, Email, License, Date Hired, Date Terminated, Training / Certification, Schedule, Type, Comments	Multiple Generic driver fields
User Interface	Delete driver	I can purge driver from system	<given> user has rights to delete driver <when>the user creates selects a driver <then> the application allows user to delete driver	All associated data, including trips, must be unscheduled. Warning message should be displayed

				prior to submitting request
Function	Make driver active	I can easily make a driver active or inactive	<given>the user has rights to create or edit driver <when>user creates new driver <then> automatically make driver active	Upon NEW make driver active. Allow user to change to inactive in future. If changed to INACTIVE, all future trip reservations should be cancelled
General Data	Add a driver picture to driver record	I can view image of driver	<given>the user has rights to create or edit driver <when>the user creates or edits driver<then> provide method to upload driver image to record	driver image may also be utilized for future mobile data app to identify driver upon boarding
General Data	Add driver capacity	I can enter the number of ambulatory and wheelchair seats the driver has	<given>the user has rights to edit driver <when>the user creates or edits driver<then> allow user to enter capacity information - 1) AMB 2) WC	
General Data	Add driver capabilities	I can assign capabilities to the driver. This tells the system what the driver is capable of operating	<given> user has rights to edit driver data <when>the user selects driver and vehicle schedule tab <then> allow user to select one or multiple driver capabilities	Multi Picklist. Values could equal - Operate Wheelchair Lift, CPR Trained, Operate Large Bus, etc.…
General Data	Assign driver to a route or run	I can assign a driver to a route for a single day or multiple days	<given> user has rights to edit driver data <when>the user selects driver and vehicle schedule tab <then>allow user to assign the driver to a defined route	NOTE: Need to define a route object
General Data	Create a driver schedule	I can create, update, or deleate a driver schedule for a defined period	<given> user has rights to edit driver data <when>the user selects driver and vehicle schedule tab <then>allow user to assign the driver schedule for a period. Days of Week, Start Time, End Time,	

1.2.4 Reservations

Reservation module allows web users or customer service representatives (CSR) to quickly and easily book trips. There are two types of trips:

- Demand Response (single trip)
- Standing Order (recurrence pattern)

COMMON ACTIONS

- NEW
- SEARCH
  - Easy and flexible search functions.
  - Once customer with trips is identified, user can edit record
    - EDIT
    - DELETE
    - COPY
- SCHEDULE / ASSIGN

Table 13: Reservation use cases

Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the reservation module	I can add, edit, or review reservations related information	<given>a user with rights to reservation module needs to launch reservation module<when>the user selects a reservation tab/button/section <then>reservation module is displayed	UI design for module accessibility will be important
User Interface	Search for reservation	Edit an existing reservation	<given>a user needs to quickly search for a single or multiple reservations <when>the user provides driver ID, phone number, name, or wildcards <then>the application provides a single or multiple customers for user to select	Search UI must be simple and fast. Wildcard searches required. Advanced search criteria required
General Data	Create New reservation	I can add a new reservation to the database	<given>a user has rights to create new reservation<when>the user selects NEW function<then>the application provides new form to enter required data	
			Trip Date	Calendar to select reservation date
			Pickup Address	Pickup Location

			Drop-off Address	Drop-off Location
			Program	Eligible Program Picklist
			Type	Pickup or Drop-off
			Time Window	Range to PU / DO customer
			Mobility Type	Defaults from customer profile
			Guests	Defaults from customer profile
			Attendants	Defaults from customer profile
			Recurrence Pattern	Recurring trip pattern. If trip has a pattern, it is considered a standing order or subscription trip
			Trip Type	Picklist of trip type
			Trip Purpose	Picklist of trip purpose
Function	Geocode addresses	I can easily geocode PU / DO addresses	<given>a user has rights to create or edit new reservation<when>the user selects address location function<then>the user can enter address details in form to find address geocode	
Function	Create a return trip	I can select a button to automatically create a return trip	<given>a user has rights to create or edit a reservation<when>the user selects RETURN TRIP function<then>the PU / DO addresses are toggled and user is asked to enter return time	
Function	Delete reservation	I can purge reservations from system	<given> user has rights to delete reservation <when>the user creates selects a reservation <then> the application allows user to delete reservation	All associated trips deleted. Warning message should be displayed prior to submitting request

General Data	Make reservation active	I can easily make a reservations active or inactive	<given>the user has rights to create or edit reservation <when>user creates new reservation <then> automatically make reservation active or inactive	Upon NEW make reservation active. Allow user to change to inactive in future. If changed to INACTIVE, all future trip reservations should be cancelled
	View trip on Map	I can easily see the pickup and drop-off on Map	<given>the user has rights to create or edit reservation <when>user selects MAP function <then> display origin destination on map	If fixed routes available, display fixed routes
	Fare Calculation	I can view estimated fare and let customer know how much to have upon pickup	<given>the user has rights to create or edit reservation <when>user selects PROGRAM associated with reservation<then> calculate the estimated fare based on the programs fare rules	Billing rules can be complicated especially for NEMT. Customer Fare and total trip cost is not necessarily the same
	Trip Comments	I can enter comments about trip that will be displayed on manifest	<given>the user has rights to create or edit reservation <when>user selects TRIP COMMENTS text box<then> allow user to enter alphanumeric text into box	Make sure you provide enough size to enter text. 264 characters' minimum.
	Confirmation Number	I can view a unique confirmation number and provide to customer on the phone	<given>the user has rights to create or edit reservation <when>user SAVES reservation <then> automatically generate a unique reservation confirmation number	
	Capacity Estimation	I can ensure that there is enough capacity to perform trip on that requested	<given>the user has rights to create or edit reservation <when>user enters required reservation data<then> automatically confirm there is capacity for the PU / DO	UI NOTE: Visual representation of capacity at time of request.
	Pickup Time Windows Estimation	I can provide the customer an estimated PU window	<given>the user has rights to create or edit reservation <when>user enters required reservation data<then> automatically display the estimated time windows	These are also called "promise" windows. On Time performance is normally calculated based on these windows

	Schedule Reservation	I can assign the reservation to an available vehicle / route	<given>the user has rights to create or edit reservation <when>user selects ASSIGN function <then> determine most efficient vehicle/route to assign to trip	Computer Assisted Scheduling. Function to bring back trip options listed from most efficient to least efficient.
	Assign to Provider	I can assign a trip to my fleet or a provider's fleet	<given>the user has rights to create or edit reservation <when>user selects ASSIGN function <then> determine most efficient provider to assign to trip	Assignment rules are a function of cost

1.2.5 Scheduling

The most complex problem associated with this application is the scheduling and routing problem. Trips must be automatically assigned to a route/vehicle pair that meets the customer requirements and does not violate system constraints.

Solves a vehicle routing problem (VRP) to find the best routes for a fleet of vehicles. A scheduler or dispatcher managing a fleet of vehicles is often required to make decisions about vehicle routing. One such decision involves how to best assign a group of customers to a fleet of vehicles and to sequence and schedule their visits. The objectives in solving such vehicle routing problems (VRP) are to provide a high level of customer service by honoring any time windows while keeping the overall operating and investment costs for each route as low as possible. The constraints are to complete the routes with available resources and within the time limits imposed by driver work shifts, driving speeds, and customer commitments. This service can be used to determine solutions for such complex fleet management tasks. The goal is to come up with an itinerary for each driver (or route) such that the deliveries can be made while honoring all the service requirements and minimizing the total time spent on a particular route by the driver.

COMMON ACTIONS

- NEW SCHEDULE
- SCHEDULE / ASSIGN OPTIMIZATION METHODS
    - Automated
    - Assisted
    - Manual
- EDIT EXISTING SCHEDULE
- CREATE ROUTE
    - CREATE RUN
        - Multiple runs can be assigned to a route
    - ASSIGN DRIVER
    - ASSIGN VEHICLE

SCHEDULE VIEWER

Scheduler requires a graphical interface to create, edit, and view:

- Schedule
- Route
- Runs
- Unscheduled Trips

- Unscheduled Runs
- Map View for route and unscheduled trip display

Scheduler requires ability to search for single routes / runs or view the entire day's schedule.

- Simple and easy to use scheduling solution that incorporates drag and drop and map based editing is recommended.

Scheduler requires ability to view daily schedules – current and in the future.

Table 14: Schedule use cases

Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the scheduling and routing module	I can create, edit, or review schedule and route related information	<given>a user with rights to scheduling module needs to launch scheduling module<when>the user selects a scheduling tab/button/section <then>scheduling module is displayed	UI design for module accessibility will be important
User Interface	Search for scheduling	Edit an existing scheduling or route	<given>a user needs to quickly search for a single or multiple vehicle/routes<when>the user provides route or vehicle information <then>the application finds route, highlights results, and zooms to extents on map	Search UI must be simple and fast. Wildcard searches required. Advanced search criteria required
General Data	Create New Schedule in Automated Mode	I can automatically insert unscheduled trips into existing schedule	<given>a user has rights to edit and create new schedules<when>the user selects SCHEDULE function<then>the system automatically inserts all or subset of trips into existing schedule	UI Note: User should be able to easily select ALL trips or filter out a subset of trips. Minimum constraints below: • Time windows • On Board Travel Time • Vehicle capacity (i.e. does vehicle have capacity at the given request time) • Vehicle capabilities (i.e. wheel chair lift)
	Optimize Routes	I can automatically route the trips in most effective and efficient manner that meets my business rules	<given>a user has rights to edit and create routes<when>the user selects SCHEDULE function<then>the system automatically creates optimized route	NOTE: This happens in conjunction with the schedule optimization function. Point to Point least cost minimum path solutions are used to generate the actual route line.

	Find Best Insertion	I can insert a single trip into an existing schedule	<given>a user has rights to edit and create schedules<when>the user selects INSERT function<then>the system automatically identifies potential insertion candidates and presents them logically to the user	Multiple options could be available and must be presented to the user
	Create route	I can create or edit an existing route	<given>a user has rights to edit and create routes<when>the user selects ROUTE EDITOR function<then>the system presents a form to create or edit a route	
	Route Name	Route Name		ROUTE EDITOR FORM
	Route Number	Number of Route		ROUTE EDITOR FORM
	Route Description	*Description of Route*		ROUTE EDITOR FORM
	Start Time	*Route Start Time*		ROUTE EDITOR FORM
	End Time	*Route End Time*		ROUTE EDITOR FORM
	Operating Days	*Days that the route operates*		ROUTE EDITOR FORM
	Assigned Vehicle	*Picklist of unassigned vehicles*		ROUTE EDITOR FORM
	Assigned Driver	*Picklist of unassigned drivers*		ROUTE EDITOR FORM
	Garage	*Picklist of Garage Locations*		ROUTE EDITOR FORM
	Create Breaks	I can insert breaks into existing route	<given>a user has rights to edit and create breaks<when>the user selects INSERT BREAK function<then>the system presents a form to create or edit new breaks for route	BREAK EDITOR FORM
		Break Name		BREAK EDITOR FORM
		Break Duration		BREAK EDITOR FORM

		Break Time Window		BREAK EDITOR FORM
		Break Location		BREAK EDITOR FORM
	Create schedule viewer	I can edit and modify schedules and routes	<given>a user has rights to edit and create schedules<when>the user selects SCHEDULE MANAGER function<then>the system presents a form where the user can select a date to create or edit a schedules that have been generated	Routes are listed with assigned vehicle and driver - Below routes are assigned trips sorted in time order (ascending). Uses will want to edit times and move trips around based on local knowledge. Grid may be considered for this or a hierarchical structure.
	Route Viewer	I can graphically view routes that have been generated on a map with route lines and stop points	<given>a user has rights to edit and create routes<when>the user selects a single or multiple routes on SCHEDULE VIEWER <then> routes and stops are graphically displayed on the MAP	UX: Simple and easy to use scheduling solution that incorporates drag and drop and map based editing is recommended. Users may want to edit or move trips on map component and quickly see impact of map. Statistics of the route should be presented (time and distance)
	Statistics Viewer	I can easily see relevant transport statistics associated with entire schedule, subset of schedule, or a single route	<given>a user has rights to open SCHEDULE MODULE<when>the user selects the daily schedule, single schedule, or multiple schedules <then> statistics are displayed on the map broken down by route	Miles Hours Trips Productivity Costs Revenue

1.2.6 Dispatch

Dispatching is the process of monitoring the performance of service delivery. Dispatchers need easy and fast access to schedule and trip information. Views or lists of routes, trips, and related performance data are required. Mapping of routes and trips is also important. Map / GIS will be used in later versions to support vehicle tracking.

DISPATCH DATA ELEMENTS

- Name
- Route
- Run

- Vehicle
- Stop Time
- Stop Type
- Stop Address
- Scheduled Time
- Time Window
- Trip Status
    - Completed
    - No Show
    - Late Cancelled
    - Cancelled with x minutes of scheduled time
        - User defined setting
    - Missed Trip
- Other Trip, Route, run data fields should be available to add to grid / layout

Calculated Fields

- Estimated Time of Arrival (ETA)
- User Interface should visually depict project late trips
- Users must be able to sort and group data in a grid or similar component

MAP

- Display scheduled routes
- Display trip origin and destination

Table 15: Dispatch use cases

Theme	I want to…	so that…	Use Case	Notes
User Interface	Access the dispatch module	I can add, edit, or review dispatch related information	<given>a user with rights to dispatch module needs to launch dispatch module<when>the user selects a dispatch tab/button/section <then>dispatch module is displayed	UI design for module accessibility will be important. Accessing information very quickly is very important
User Interface	Select a date to dispatch	I can monitor and manage the status of routes, trips, performance, schedule for the day	<given>a user needs view dispatch screen <when>the user selects a calendar function <then>the user selects the date that they wish to perform dispatch functions	Dispatch should default to current date
User Interface	Search for data	I can easily find customers, trips, and routes to view status	<given>a user needs to quickly search for a single or multiple data elements <when>the user provides name, confirmation number, route, vehicle, or wild card <then>the system filters all data elements that meet criteria	Fast and easy search functions are important

User Interface	View schedules and trips in grid	I can easily view a grid with related schedule information	<given>a user has permission to view dispatch <when>the user open dispatch form<then>the application provides easy to view grid (or similar component) to view scheduled data elements	Grid or similar component can be used for dispatch list view
	Edit schedule and trip data	I can edit scheduled data elements to reflect actual performance	<given>a user has permission to view and edit dispatch <when>the user selects a record<then>the application allows user to update record	Need to maintain scheduled information and actual information separately for comparisons (i.e. He was scheduled to be picked up at 11:30; Actual pickup was 12:15.
	View dispatch data elements	I can view relevant schedule information in dispatch view	<given>a user has permission to view and edit dispatch <when>the user open dispatch form<then>the application displays following data elements (minimum… there are probably more)	UX: Allow users to select and organize columns to include in dispatch view
		Customer Name		
		Route		
		Driver		
		Vehicle		
		Scheduled Stop Time		
		Actual Stop Time		
		Mobility Type		
		Trip Type		
		Trip Purpose		
		Program		
		Type (PU / DO)		
		Stop Address		
		Stop City		

		Early Time Window		
		Late Time Window		
		Trip Status	Cancel, Late Cancel, No Show, Completed	
		Route Pullout Time		
		Route Pulling Time		
		Trip ETA	Calculated field	
		Route ETA	Calculated field	
	See routes on Map	I can determine status and location of trips	<given>a user has permission to view dispatch <when>the user selects a MAP function<then>a map with scheduled routes and stops is displayed on map	Map control must have common map tools such as: Zoom in, Zoom Out, Pan, Find Address
	Sort and Group	I can sort and group schedule by multiple methods	<given>a user has permission to view dispatch <when>user selects dispatch data element header<then>column can be sorted and grouped by multiple methods and levels	UX Note: Provide ability to save grid layouts for future use (i.e. Late Trips, OTP)

1.2.7 Analytics

Reporting and data analytics is an extremely important component of the system. In fact, it could be the biggest. Systems and users must be able to access all of the data in the system via standard and ad hoc reports.

STANDARD REPORT THEMES

- Major Object Reports
  - Customers
  - Drivers
  - Reservations
  - Schedule
  - Dispatch
- Operations
- On Time Performance
- Productivity
- Financial

- Exception

Samples:

- Driver Manifest
- Route Summary
- Schedule Productivity
    - Route
    - Run
    - Driver
- Cancellation and No Shows
- Operating Statistics
- Exception Reports
- Financial Invoices
    - Cost Allocation by Program

Table 16: Report use cases

Theme	I want to…	so that…	Acceptance Criteria	Notes
User Interface	Access the report module	I can add, edit, or review dispatch related information	<given>a user with rights to dispatch module needs to launch dispatch module<when>the user selects a dispatch tab/button/section <then>dispatch module is displayed	UI design for module accessibility will be important. Accessing information very quickly is very important
User Interface	Search for reports	I can easily find reports	<given>a user needs to quickly search for a single or multiple data elements <when>the user provides name, confirmation number, route, vehicle, or wild card <then>the system filters all data elements that meet criteria	Fast and easy search functions are important
User Interface	Access reports	I can easily run reports that I have permissions to access	<given>a user needs to run a report<when> the user searches or selects a report <then> the report is run and displayed	Administration must provide ability to set permissions at the report level: CRUD
User Interface	Run reports with parameters	I can easily select different parameters for the report	<given> the user wants to run a report with valid permissions<when> user selects report <then>a parameter form is presented that allows user to enter various report parameters and pass into report	Common Parameters: Date Range, Group By, Sort By, Filter By, Sum, Average

User Interface	View Reports in Folders	I can easily organize reports by functional area	<given>user has permissions to access report module <when> user selects NEW FOLDER function <then> system allows user to create a folder under PERSONAL REPORTS	Standard Report Objects: Customers, Drivers, Operations, Productivity, Financial, Exceptions, Dashboards
User Interface	Create custom reports	I can easily create my own reports and save them to public or private folders	<given>user has permissions to create ad hoc reports <when> user selects NEW REPORT <then>NEW REPORT FORM is displayed that allows user to create custom report with desired fields, grouping, sorting, filters, graphs, pivots, and logic	UX: Major differentiator opportunity.
User Interface	Export reports	I can easily email or export to other formats	<given> user has permission to run reports<when>user runs selected report<then>user has options to export or email to standard file formats	CSV, PDF are two most common
User Interface	Standard Reports	I can easily view common and industry standard reports	<given> user has permission to run reports<when>user can easily find and select reports<then>user can run desired standard reports with various parameters	Users cannot SAVE standard reports. Only SAVE AS to Personal Workspace

6. Detailed Design

Instructions: Provide the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software components, and interconnect the hardware and software segments into a functional product. Additionally, address the detailed procedures for combining separate COTS packages into a single system.

ARC will continue to utilize third party virtualization services for its physical hardware, internet provisioning, security, and platform hosting. ARC currently utilizes both Heroku and AWS to fill this requirement.

The current is an open source application. The proposed solution may be open source, proprietary or a combination of both. Integration between the current solution and a COTS solution, if available and selected, will occur at the API level using a RESTful framework.

6.1 Hardware Detailed Design

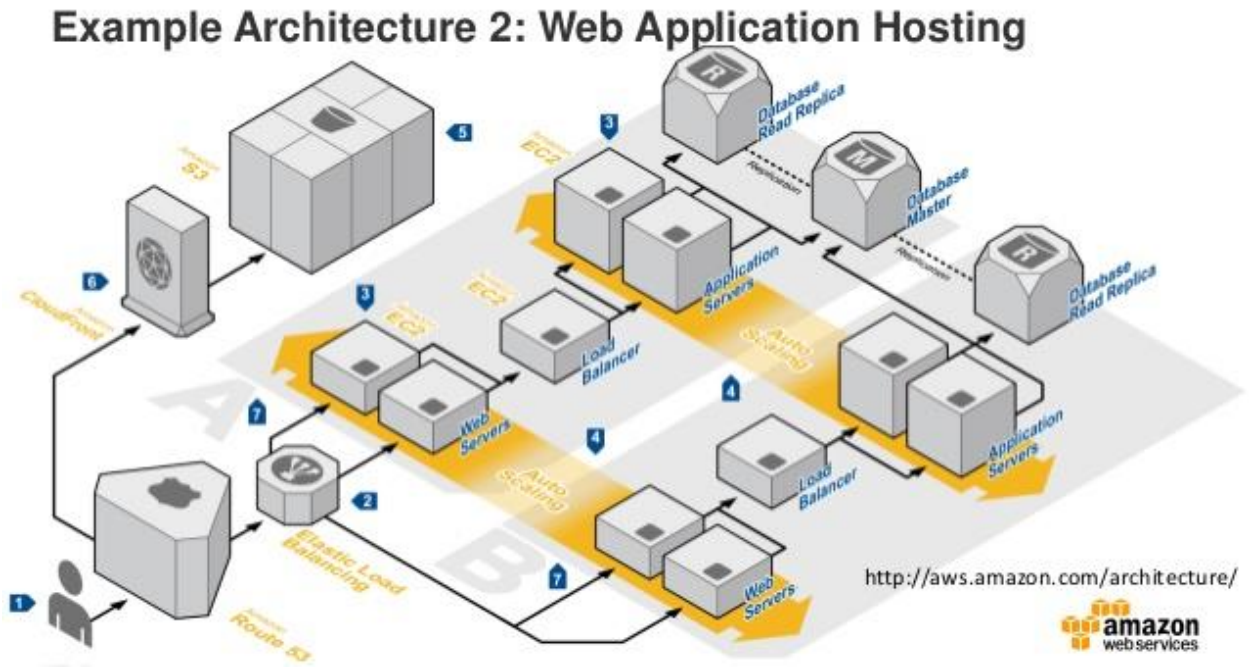


Figure 12: Example of web application hosting

6.2 Software Detailed Design

Provide a detailed description for each system software service that addresses the following software service attributes. Much of the information that appears in this section should be contained in the headers/prologues and comment sections of the source code for each component, subsystem, module, and subroutine. If so, this section may largely consist of references to or excerpts of annotated diagrams and source code. Any referenced diagrams or source code excerpts should be provided at any design reviews.

- *Service Identifier - The unique identifier and/or name of the software service*
- *Classification - The kind of service (e.g., application, data service, etc.)*

- *Definition - The specific purpose and semantic meaning of the service*
- *Requirements - The specific functional or nonfunctional requirements that the service satisfies*
- *Internal Data Structures - The internal data structures for the service*
- *Constraints - Any relevant, assumptions, limitations, or constraints for the service (this should include constraints on timing, storage, or service state, and might include rules for interacting with the service (encompassing pre-conditions, post-conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.))*
- *Composition - A description of the use and meaning of the subservices that are a part of the service*
- *Users/Interactions - A description of the service's collaborations with other services (what other services use this this entity? what other services does this entity use (including any side-effects this service might have on other parts of the system)? this includes the method of interaction, as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated sub-classes, super-classes, and meta-classes)*
- *Processing - A description of precisely how the service goes about performing the duties necessary to fulfill its responsibilities (this should encompass a description of any algorithms used; changes or state; relevant time or space complexity; concurrency; methods of creation, initialization, and cleanup; and handling of exceptional conditions)*
- *Interfaces/Exports - The set of services (resources, data types, constants, subroutines, and exceptions) that the service provides (the precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc.; for each service element described, include or provide a reference in its discussion to a description of its important software service attributes (Component Identifier, Classification, Language, Source Lines of Code (SLOC) Estimate, Definition, Responsibilities, Requirements, Internal Data Structures, Constraints, Composition, Uses/Interactions, Resources, Processing, and Interfaces/Exports))*
- *Reporting Design and Integration - If built in, provide details on data traffic and volumes*

6.3 Security Detailed Design

Instructions: Provide a graphical representation with detailed information for each of the individual security hardware components. Specify the design for the below items as required.

- *Authentication*
- *Authorization*
- *Logging and Auditing*
- *Encryption*
- *Network ports usage*
- *Intrusion Detection and Prevention*

The design should be based on the designated system security level and provide adequate protection against threats and vulnerabilities.

Current and proposed application utilizes AWS IAM for security and authentication.

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources for your users. You use IAM to control who can use your AWS resources (*authentication*) and what resources they can use and in what ways (*authorization*).

The "identity" aspect of AWS Identity and Access Management (IAM) helps you with the question "Who is that user?", often referred to as authentication. Instead of sharing your root account credentials with others, you can create individual IAM users within your account that correspond to users in your organization. IAM users are not separate accounts; they are users within your account. Each user can

have its own password for access to the AWS Management Console. You can also create an individual access key for each user so that the user can make programmatic requests to work with resources in your account. In the following figure, the users Brad, Jim, DevApp1, DevApp2, TestApp1, and TestApp2 have been added to a single AWS account. Each user has its own credentials.

If you are creating a mobile app or web-based app that can let users identify themselves through an Internet identity provider like Login with Amazon, Facebook, Google, or any OpenID Connect (OIDC) compatible identity provider, the app can use federation to access AWS.

Amazon Cognito should be used for proposed solution security improvements. Amazon Cognito lets you easily add user sign-up and sign-in and manage permissions for your mobile and web apps. You can create your own user directory within Amazon Cognito, or you can authenticate users through social identity providers such as Facebook, Twitter, or Amazon; with SAML identity solutions; or by using your own identity system. In addition, Amazon Cognito enables you to save data locally on users' devices, allowing your applications to work even when the devices are offline. You can then synchronize data across users' devices so that their app experience remains consistent regardless of the device they use.

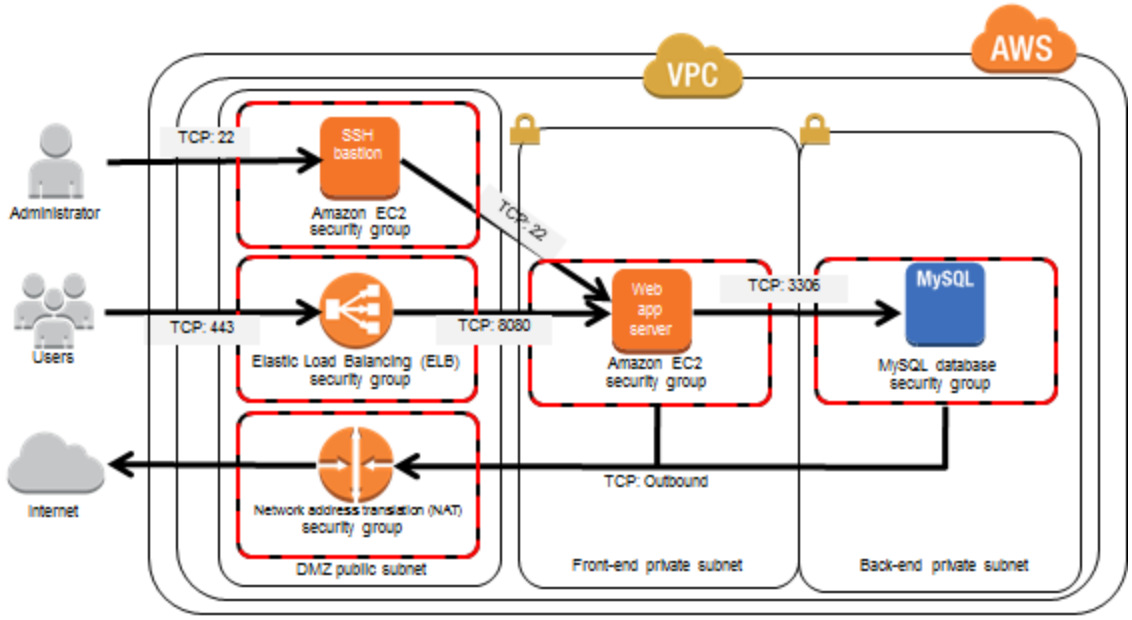


Figure 13: Security detail design

6.4 Performance Detailed Design

The AWS platform used by the current solution provides elastic and on demand capacity and availability. Performance management and monitoring is integrated into solution. Hardware architecture scales as needed based on demand. There are no single points of failure in current and proposed data center architecture.

6.5 Internal Communications Detailed Design

The proposed solution utilize existing communication protocols and methods. There are no additional components, servers, or applications to communicate with. The proposed solution will simply extend the current capabilities.

7. System Integrity Controls

The following section documents security integrity controls.

- Internal security to restrict access of critical data items to only those access types required by users/operators
    - The current system is roles and permission based. Form and field level security will be available in the proposed system. Based on user roles and permissions, system administrators can control data access by user and provider.
- Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
    - Audit logs and audit trail is proposed in the current use cases. All changes to data must be recorded by date, time, and user. Data will be managed indefinitely or until a system admin purges the data.
- Application audit trails to dynamically audit retrieval access to designated critical data
    - See above.
- Standard tables to be used or requested for validating data fields
    - Industry standard data validation rules, triggers, and process have been identified in this document.
- Verification processes for additions, deletions, or updates of critical data
    - Verification of additions, deletion, updates, etc… are controlled in the data validation functions described in another section.
- Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.
    - Data audit trails and logging will provide detailed audit of all data. Network and application logs identify network terminations, user terminations, bugs, and catastrophic failures. These allow system admin to improve the application and hosting environment.

8. External Interfaces

A number of services external to the 1-Click software can be used to provide additional fixed-route, taxi, or ride share information to travelers.

8.1 ATL Transit

ARC has created and maintains ATLTransit, a regional transit information hub web application for transportation options not currently offered by existing trip planning applications. ATLTransit contains route information from all of the transit agencies in the region. ATL Transit is a regional fixed route trip planner that leverages both Google Maps and Open Trip Planner. SGT connects to this system for fixed route planning information. It is integrated to SGT via API.

8.2 Google Maps

Services from Google are used to display the background of the maps within 1-Click, and to geocode street addresses to determine full addresses and the latitude/longitude location.

As such, a unique Google Maps API Key is needed for each 1-Click instance.

8.3 OpenTripPlanner

OpenTripPlanner provides fixed-route transit, walking, bicycling, and driving itineraries for the SGT system. OpenTripPlanner requires GTFS data from each public transit agency. OpenTripPlanner also requires a graph of the street network derived from OpenStreetMap.

An instance of OpenTripPlanner will be setup and maintained as part of the proposes system.

8.4 Rideshare

Rideshare integration is not available due to the propriety nature of these services, no standard integration method can be implemented. If a public API is available, the proposed system may be able to integrate this mode into system.

8.5 Taxi Fare Finder

SGT uses a taxi fare estimator known as Taxi Fare Finder to estimate taxi fares for trips. Taxi Fare Finder does not require any additional hardware or services to be setup. Taxi Fare Finder provides a public API that the system will query as part of the trip planning process.

8.6 Transportation Network Companies (TNC)

TNC's may be implemented in the proposed system. Uber and Lyft both provide well published open API's for integration into third party systems. The system would be able to take advantage of these business models to complement or supplement transportation services.

8.7 GTFS Real Time

Regional transit systems may publish real time GTFS feeds. This allows third party systems to consume the real time location data for fixed route buses and rail. MARTA currently publishes this information on their developer website. Other regional transit systems may choose to do this in the future. If so, this dramatically improves the same day trip planning and multi-modal coordination capabilities of these types of systems.

Sample MARTA Real Time GTFS Feed Response:

[{ "ADHERENCE": "4", "BLOCKID": "31", "BLOCK_ABBR": "110-4", "DIRECTION": "Northbound", "LATITUDE": "33.8346347", "LONGITUDE": "-84.3824637", "MSGTIME": "5\/14\/2013 11:14:04 AM", "ROUTE": "110", "STOPID": "900456", "TIMEPOINT": "Peachtree Hills & Peachtree", "TRIPID": "3719918", "VEHICLE": "2853" }, ...]

```
{
  "request_id": "852b8fdd-4369-4659-9628-e122662ad257",
  "product_id": "a1111c8c-c720-46c3-8534-2fcdd730040d",
  "status": "processing",
  "vehicle": null,
  "driver": null,
  "location": null,
  "eta": 5,
  "surge_multiplier": null
}
```

Name	Type	Description
request_id	string	The unique ID of the Request.
product_id	string	Unique identifier representing a specific product for a given latitude & longitude. For example, uberX in San Francisco will have a different product_id than uberX in Los Angeles.
status	string	The status of the Request indicating state.
vehicle	object	The object that contains vehicle details.
driver	object	The object that contains driver details.
location	object	The object that contains the location information of the vehicle and driver.
eta	integer	The estimated time of vehicle arrival in minutes.
surge_multiplier	float	The surge pricing multiplier used to calculate the increased price of a Request. A multiplier of 1.0 means surge pricing is not in effect.

Figure 14: Sample uber API responses

8.8 GTFS Flex

GTFS flex is the newest emerging transit standard that models mobility on demand, demand response, and flex-deviation services. As an example, CCT has recently implemented a flex service that could be integrated into this system.

8.9 Emerging Business Models

Integration and connectivity to emerging business models must be envisioned in the TCMP. Car sharing, bike sharing, and other emerging models should be supported via published and open API's.

8.10 Third Party Commercial Application Integration

Many providers have implemented third party applications for customer management, eligibility, reservations, scheduling and dispatching. Integration and connectivity support may be required to coordinate and exchange data across multiple platforms. It is anticipated that an open and published set of API's will facilitate this effort. Current research projects in Portland, Oregon, are exploring the concept of open sourced transportation clearinghouses to support this.

Major commercial providers include:

- Trapeze
- RouteMatch
- SimpliTransport
- Ecolane

- Mobilitat
- Stratagen

8.11 Transportation Clearinghouse

The Ride Connection Clearinghouse ("Clearinghouse") is a web site that allows ride services to share trips that cannot be fulfilled and claim trips shared by other services. The Clearinghouse API is an Internet-accessible programming interface that allows services and third parties to integrate other software with the Clearinghouse to automate the sharing and claiming of trips. One such system has already been developed: the Ride Connection Clearinghouse Adapter. This reference is intended to assist with further work on the Adapter as well as the development of new adapter software.

8.11.1 Adapter API

The Ride Clearinghouse Adapter is a software system that simplifies back office integration with the Ride Clearinghouse web site. The Adapter runs as a Windows Service in the background, periodically triggering a worker process that synchronizes data with the Clearinghouse API, then imports new data from a user's system to send to the Clearinghouse. The Ride Clearinghouse web site supports manual import and export (upload and download) of trip tickets via the Bulk Upload menu. Imported trip tickets must be formatted as text files in the CSV format.

8.12 Points of Interest

Trip origins and destinations can be located interactively on a map, using the GPS of a mobile device, via the Google geocoder that will convert a street address into a X-Y coordinate, and by selecting from a pre-defined list of Points of Interest (POIs). These POIs often include hospitals, schools, offices of Veterans Affairs, and other popular locations.

In order to include points of interest in the 1-Click database, a comma separated value (CSV) file is required containing the list of POIs. For each point of interest, the following fields are needed.

Table 17: 1-Click Points of Interest File Format

Field	Type	Description
LONGITUDE	Double	Longitude (X) coordinate of the location
LATITUDE	Double	Latitude (Y) coordinate of the location
NAME	String	Name
ADDRESS_1	String	Address (line 1)
ADDRESS_2	String	Address (line 2)
CITY	String	City name
STATE	String(2)	State abbreviation
ZIP	String(5)	5-digit zipcode
COUNTY	String	County name
TYPE (Optional)		

8.13 Public Transit

Multiple public transit agencies may be operating within the 1-Click area. Each one needs to exist within the 1-Click database, together with the defined fixed service routes.

8.14 GTFS

The General Transit Feed Services[1] (GTFS) defines a common format for public transportation schedules and associated geographic information. These GTFS files are loaded into 1-Click and are consumed by the OpenTripPlanner server to generate public transit trip itineraries.

8.15 Agencies

Table 18: Public Transit Agency Attributes

Attribute	Description
Name	Name of the Public Transit Agency
Street	Contact Address
City	
State	
Zipcode	
Phone	General phone number for Public Transit Agency
Email	General information email address for Public Transit Agency
Website	URL of the Public Transit Agency website
Logo	Image of the Public Transit Agency logo suitable for display within 1-Click
Administrator	Name of the person who will be the 1-Click Public Transit Agency Administrator. Administrator needs a 1-Click user account prior to being assigned.

Note: Some of these attributes can be extracted from the GTFS file.

8.16 Specialized Service Providers

Demand-Response Services are provided by a number of Providers.

8.17 Providers

The following data is required about each Provider.

Table 19 - Specialized Services Provider Attributes

Attribute	Description
Name	Name of the Provider
Street	Provider Address
City	
State	
Zipcode	
Phone	General phone number for Provider
Email	General information email address for Provider
Website	URL of the Provider website

Logo	Image of the Provider logo suitable for display within 1-Click
Administrator	Name of the person who will be the 1-Click Provider Administrator. Administrator needs a 1-Click user account prior to being assigned.

8.18 Services

The following data is required about each Service offered by a Provider.

Table 20: Provider Services attributes

Attribute	Description
Provider Name	Name of the Provider
Service Name	Name of the Service
Contact Name	Contact details of the person responsible for this service
Contact Phone	
Contact Email	
Schedule	Daily start and end times of the services
Advanced Booking	How much advanced notice is required to book a trip with this service?
Accommodations	What traveler accommodations are provided for this service?
Eligibility Requirements	What are the eligibility requirements to use this service?
Trip Purposed Served	Is this service only for specific trip purposes? If so, what trip purposes?
Service Area	The geographic area covered by this service – can be split into Origin, Destination and Resides areas.
Fare Information	Fare structure for this service

As part of the 1-Click trip planning workflow, users need to define the start and end of the trips and display information (e.g., start/end locations, sidewalk obstructions, trip routes, etc.) on a map. To accomplish this, 1-Click uses a number of third party application programming interfaces (API):

- Google Geocoding API[2] for geocoding addresses, and reverse geocoding point locations.
- Leaflet Map API[3] for rendering maps.
- Google Street View Image API [4]for displaying a street view of a given location.

The pros and cons regarding the use of these APIs is discussed below with potential alternatives.

8.18.1 Geocoding

1-Click allows users to define the start and end locations of a trip by:

1. Entering a street address which is then geocoded to determine a longitude/latitude.
2. Selecting a predefined Point of Interest that has an associated longitude/latitude.
3. Clicking on the map that returns a longitude/latitude that is reverse geocoded to determine the street address at that location.

To implement methods 1 and 3, the Geocoding API must allow both forward and reverse geocoding, and be available as a web service that can be called directly from 1-Click.

There are many Geocoding APIs that could potential be used within 1-Click (e.g., Gisgraphy, Google, Here, MapQuest, Nominatim, Yahoo!). While any of these Geocoding APIs could potentially be used within 1-Click, the strength of the geocoder is based on the quality of the returned geocodes. Based on research, third-party review and comparison testing, the Google Geocoder was selected due to the quality of the returned geocodes in comparison to the competitors.

CS will be updating the geocoding implementation within 1-Click v1.2 to use client-side geocoding instead of server-side geocoding to increase the number of daily free geocodes.

8.18.2 Maps

1-Click displays maps when defining the start and end points of a trip; displaying the trip itinerary routes; defining Traveler Places; and displaying Provider Service coverage areas. To do this, the select Map API must be able to:

- Zoom to a specific map extent (e.g., area around start/end of the trip, current location, extent of the route, etc.).
- Display a map background, known as a basemap, that provides context.
- Render various graphics (e.g., start/end trip locations, trip itinerary routes, other spatial data).
- Allow users to interact with the map (e.g., zoom/pan, identify graphics, determine clicked locations, etc.).

Based on these requirements, there are a number of Map APIs that could potential be used within the proposed solution. The pros and cons of these different Map APIs are listed below.

Table 21: Map API pros & cons

Map API	Pros	Cons
Google	• Widely used • Extensive set of graphic controls	• Proprietary • Detailed license language • HTML4
Leaflet	• Open-source • Light-weight (i.e., small download size that improves performance) • Fully HTML5 compatible • Seamless integration with multiple formats of basemap (e.g., Esri, Google, OpenStreetMap) • Seamless integration with multiple formats of feature map service (e.g., Open Geospatial Consortium (OGC) WFS and WMS, Esri) • Extensive set of graphic controls	
OpenLayers	• Open-source • Seamless integration with multiple formats of basemap (e.g., OpenStreetMap, Bing, MapQuest)	• HTML4

The Leaflet API was selected for displaying maps in 1-Click due to its high level of functionality, compatibility with a modern open source HTML5 application, and flexibility to show different basemaps.

The Google Maps API license that states that Google Geocodes must be displayed on a Google basemap. As such, the Leaflet implementation within 1-Click uses a plugin[5] that uses the Google Maps API v3 to display the Google basemap. CS believes that 1-Click complies with the Google Maps license, and will take responsibility as part of Maintenance and Support services for resolving any claims by Google to the contrary.

8.18.3 Google Street View

Google Street View pages are displayed by passing the longitude/latitude coordinates as URL parameters to the Google Street View Image API.

Use of Google Street View is considered separate from the use of other Google Map APIs, with the license stating "As another example, you must not display Street View imagery alongside a non-Google map, but you may display Street View imagery without a corresponding Google map because the Maps APIs Documentation explicitly permits you to do so."

8.19 Interface Architecture

The current and proposed solutions utilize a services oriented architecture. The proposed system will utilize the existing interface architecture by implementing a REST framework.

Representational State Tranfser (REST)

This is a style or framework for designing integrated applications or services over HTTP. The proposed solution will implement a true RESTful API for interapplication integration and regional coordination. This achieves the following results from an interface architecture perspective:

- Uniform interface
- Client–server
- Stateless
- Cacheable
- Layered system
- Code on demand

Data Exchange

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for developers to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

8.20 Interface Detailed Design

All third party applications and integrations will utilize a RESTful API that will be designed in the application development phase. The API will be open and published. This means that any developer or

third party, if provide the proper security credentials, can access application, database, and published functions.

REST Model Clients make standard HTTP requests over an SSL channel and should always validate the certificate of the endpoint with which the client is communicating.

Resources

Clients make requests against enStratus resources, either in aggregate or a specific resource.

The format of the URL is:

- endpoint/version/namespace/resource[?query_parameters] • endpoint/version/namespace/resource/resource_id

Operations

API requests are standard HTTP requests against SGT published resources. GET queries for a list of a class of resources or the details of a specific resource. POST creates a new resource instance and will provide either a job or a resource tance in the response body. PUT updates an existing resource with the specified parameters. DELETE removes or terminates or deactivates a resource. In general, there is no such thing as permanent deletion of anything in enStratus. Resources are instead “deactivated” via the DELETE call. HEAD provides response headers, including a count of matching resources.

Request Headers

When making a request, there are three authentication headers the third party must specify plus an optional authentication header for preventing replay attacks.

In addition, the API must support the following optional request headers:

- Accept
- x-es-details
- x-es-with-perms

Third parties may specify an “Accept” header to define whether you wish to receive responses as XML or JSON. The default response is XML. The values you may specify for “Accept” are:

- application/xml
- application/json

Response Codes

API will respond with standard HTTP response codes appropriate to the result of the request. While the exact meaning of the code varies depending on the request, the general rules are:

200

A response code of 200 means the request was successful and details about the response can be found in the body of the response.

201

The requested POST operation was successful and an object was created in the system.

202

The requested operation has been accepted and the body contains information about an

asynchronous job you can query to check on the progress of the request.

204

The requested operation was successful and there is no response body.

307

Please repeat the request using the provided URI. Subsequent requests can use the old URI.

400

Your request was improperly formatted. You should verify that your request conforms to this specification and re-issue the request in a properly formatted manner.

404

The requested resource does not exist.

409

An operational error occurred. The most common reason is an error with the cloud provider itself, but it can also result from any number of cloud state issues.

418

A request was made to create a resource, but the resource was not created and no job was returned.

500

API failed to process the request because of an error inside the system.

501

You requested an action against a resource in a cloud that does not support that action.

503

API undergoing maintenance or is otherwise temporarily unavailable for API queries.

Response Entities

All GET methods respond with the JSON or XML of the resource(s) being requested. HEAD methods have no response entity.

POST methods may respond with a 201 CREATED or 202 ACCEPTED response code depending on whether the creation completed immediately or is an asynchronous operation. If the resource was created immediately, API should provide a JSON or XML entity that includes the new resource's unique ID. If the creation operation takes time, however, the response body will include a Job resource that can be tracked to completion.

PUT and DELETE methods generally respond with 204 NO CONTENT unless the operation is a long-lived operation. In those scenarios, the PUT will respond with a 202 ACCEPTED response code and include a Job resource in the response entity

9. Appendix A: Record of Changes

Table 22: Record of changes

Version Number	Date	Author/Owner	Description of Change
1.0	*02/26/2017*	*Tim Quinn*	*Draft document*
2.0	*03/12/2017*	*Carly Harper*	*Revisions*
3.0	*04/26/2017*	*Cyndi Burke*	*Feedback / Process*
4.0	*5/02/2017*	*Tim Quinn*	*Final Draft*
5.0	*05/05/2017*	*Cyndi Burke*	*Review*
6.0	*05/05/2017*	*Carly Harper*	*Modifications*
7.0	*05/08/2017*	*Tim Quinn*	*Final Draft (v1)*
8.0	*06/22/2017*	*Tim Quinn / Carly Harper*	*Comments from FTA / Modifications*