All-En-Route Software Architecture

1. All-En-Route software overview

All-En-Route software will logically follow layered architecture supporting a scale of millions of users using the All-En-Route user applications in the presentation layer, served by multi-site geographically replicated business and persistence layers (core services and data store), supporting all the required quality attributes listed in the “All-En-Route Functional Architecture and Architecture Principles” document (referred to as “Functional Architecture” from this point onward).

<table>
<tr><td>Presentation Layer
[All-En-Route user applications]</td><td colspan="2">Rider application</td><td colspan="2">Driver application</td><td colspan="2">Operations application</td></tr>
<tr><td>Business Layer
[All-En-Route core business logic]</td><td>User management service</td><td>Mapping service</td><td>Ride management service</td><td>Payment processing service</td><td>Food-delivery processing service</td><td>Analytics service</td></tr>
<tr><td rowspan="2">Persistence Layer</td><td colspan="6">Data store service</td></tr>
<tr><td>User and Cab data store</td><td>Ride data store</td><td>Payment data store</td><td colspan="2">Analytics data store</td><td>Operations data store</td></tr>
</table>

Fig 1: All-En-Route high level architecture layers

The presentation layer will consist of user applications for passenger (rider), cab-provider (driver) and operations. The business layer will consist of core services for business logic and persistence layer will consist of data store (management) service for all the data stores. The User and Cab data store will be contextually referred to as User data store or Cab data store in the document. The data stores will be replicated to designated centralized site locations for backup and disaster recovery reasons. Analytics data stores will also be operated only in such central locations to allow for full data access for the analytics across all sites. The Operations data store will have customer issues and knowledge base data. The knowledge base data will also be replicated and merged across all sites for optimal issue resolutions.

2. All-En-Route user applications

All-En-Route will have separate **applications for riders** (used by passengers and kiosk operators), **drivers** (used by cab-providers) and **operations** (used by business owner, system administrator, customer and techinical support personnel). The applications will mainly constitute **presentation layer** and **minimum necessary business rules processing** with **no data or operational state** being **persisted locally**.

2.1 Rider Application

The functionality available in this application will be controlled by role based access control (**RBAC**). The **roles** will include **passenger** and **kiosk operator**. It will be available as **mobile application for passengers** and **web application for kiosk operators**.
It will provide **client side interface** for **account management, ride management, location tracking, food-delivery order and payments**.

2.2 Driver Application

This will be available as **mobile application for drivers (cab providers)**. It will provide **client side interface** for **onboarding, authentication, ride management, location tracking and payments**.

2.3 Operations Application

The functionality available in this application will be controlled by role based access control (**RBAC**). The **roles** will include **system administrator, customer support personnel, technical support personnel** and **business owner.**
For **system administrator** role, it will be available as **web application** to provide **interface for user management, service and platform configuration, install, uninstall and upgrades**.
For **customer support personnel** and **techinical support personnel** roles, it will be available as **web application** to provide **interface for customer issues management and knowledge base management operations**.
For **business owner** role, this will be available as **web application** that runs on laptop **as well as mobile application** that runs on mobile device. It will provide **interface for analytics functionality** for business owner.

2.4 Application level forward traceability

Functionality **[Ref: Sections 1 to 7 in Functional Architecture document]**	**All-En-Route Application** **[Ref: Sections 3 to 5 in this document]**
1. Functionality for passengers	3. Rider Application
7. Functionality for kiosk operators	
2. Functionality for cab providers	4. Driver Application
3. Functionality for cab aggregator (business owner)	5. Operations Application
4. Functionality for system administrator	
5. Functionality for customer support personnel	
6. Functionality for technical support personnel	

3. Rider Application

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

3.1 Application function level forward traceability

Functionality **[Ref: Sections 1 and 7 in Functional Architecture document]**	**All-En-Route Rider Application** **[Ref: Section 3 in this document]**
1. Functionality for passengers	3. Rider Application
1.1 Account management	3.2.1 Account management interface
1.1.1 Registration	3.2.1.1 Signup
1.1.2 Authentication	3.2.1.2 Login and logout
1.1.3 Profile Management	3.2.1.3 Profile
1.2 Mapping services	3.2.2 Map interface
1.2.1 Locations and routes	3.2.2.1 Map interface (Source, destination and route)
1.2.2 Location tracking	3.2.2.2 Map interface (Periodic cab location update)
1.3 Ride management (and 7.1 Ride booking)	3.2.3 Ride management interface
1.3.1 Ride request creation	3.2.3.1 Ride request creation and assignment
1.3.2 Ride assignment and notification	3.2.3.2 Cab assignment notification
1.3.3 Ride cancellation	3.2.3.3 Ride cancellation
1.4 Enroute food delivery	3.2.4 Food delivery interface
1.4.1 Food stops and options listing for ride route	3.2.4.1 Food stops and options listing for ride
1.4.2 Food order delivery functionality for ongoing ride	3.2.4.2 Food order for ride
1.5 Payments management	3.2.5 Payments interface
1.5.1 Ride payments	3.2.5.1 Ride payment
1.5.2 Food delivery order payments	3.2.5.1 Ride payment (includes food order payments)

3.2 Application modules and functional flows

The application will have following **modules**:

- Account management interface
- Map interface
- Ride management interface
- Food delivery interface
- Payments interface

3.2.1 Account management interface

This module will provide interface for **registration, authentication and profile management** functionality described in “Section 1.1. of Functional Architecture” document.

3.2.1.1 Signup

Signup or self registration will be available to passengers only, kiosk operator accounts will be predefined by system administrator. The passenger will use registration UI screen to enter registration details. **Mandatory input** checks and **standard input format validations** will be performed on the information entered by passenger. Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security.
It will invoke core service entry points to verify passenger details and then to persist verified passenger details. On success, user will be registered user and can login and start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt registration again.

3.2.1.2 Login and logout

The passenger or (kiosk) operator will use login UI screen to log into the application. **Role (passenger or kiosk operator), UserID and password inputs** will be provided by user on login screen. The **password** will be **one-way-hashed**. The user entered credentials will be verified against persisted credentials in the User data store using core service entry point. On successful verification, user will be logged into the application and can start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt login again with a maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

3.2.1.3 Profile

Once successfully logged in, user can choose to update the profile information. Once user confirms updates to the profile, **PII** information in the profile details will be **encrypted**. The user entered profile updates will be persisted in the User data store using core service entry point and status will be communicated to the user.

3.2.2 Map interface

This module will provide interface for **location, location tracking and ride routes** functionalities as described in “Section 1.2 of Functional Architecture” document.

3.2.2.1 Map interface (Source, Destination and Route)

The user will be able to see the **map view** and provide source (pickup) and destination (drop-off) **locations**. The recent (upto 10) location searches will be displayed. The selected locations will be displayed as markers on the map. Once user confirms the locations, the **route** information will be displayed on the map view.

3.2.2.2 Map interface (Periodic cab location update)

Once the ride is confirmed, the user will be able to see **live location of the assigned cab** on the map view. The live location will be available throughout the ride. The user will also be able to **share** the **live location** for user safety purposes.

3.2.3 Ride management interface

This module will provide interface for **ride request, cab assignment notification and ride cancellation** functionalities as described in “Section 1.3 of Functional Architecture” document.

3.2.3.1 Ride request creation and assignment

The user will be able to view the **ride options** - all the available cab types and price for the type of cab for the given route. The user can select one cab type and proceed with setting ride time. Ride time can be set to now or schedule for later. The **user selected ride option** will be **persisted** in the Ride data store as ride request using core service entry point.

3.2.3.2 Cab assignment notification

The user will be **presented with status of cab assignment**. The interface will get push notification or invoke core service entry point to get assigned cab and driver details periodically. **Once** the ride is **assigned**, it will present the ride information received (**ride OTP, details of cab and driver) to the passenger**. The ride will be maintained in the **upcoming rides** for the passenger before start of the ride. The ride assignment changes (regarding the assigned cab and driver, if any) will be push notified or queried periodically using core service entry point and presented to the user.

3.2.3.3 Ride cancellation

User will be allowed to **cancel** the **upcoming ride** request before the ride assignment by clicking on the “Cancel Ride” option available on the UI screen. The cancellation will result in invocation of core service entry point to handle cancellation processing according to the logic described in “Section 1.3.3 of Functional Architecture” document.

3.2.4 Food delivery interface

This functionality will be available **only in case of long-distance rides**. It will provide **food stops and food options listing** for the upcoming or ongoing ride and **food order functionality** as described in “Section 1.4 of Functional Architecture” document.

3.2.4.1 Food stops and options listing for ride

It will invoke the core service entry point to get **list of all the available food stops on current ride route** as per the logic described in “Section 1.4.1 of Functional Architecture” document. User will be able to see the food stops list and once the user selects a food stop, it will provide the list of available food options using core service entry point.

3.2.4.2 Food order for ride

The user will be able to select from **food options presented for selected food stop**. Once selected and confirmed, it will invoke core service entry point to process and persist the food order and present the status to the user.

3.2.5 Payments interface

This module will provide interface for **ride (and food bill, if any) payment functionality** as described in "Section 1.5 of Functional Architecture" document.

3.2.5.1 Ride payment

Once the ride is marked finished by driver, the ride payment amount will be fetched using core service entrypoint. The ride payment interface will allow user to make the ride payment. User will be provided with an interface with **digital payment options**: Internet banking, Debit/Credit card and UPI. After selection of the payment method, interface for payment authentication will be presented. The **payment details entered** by user will be **masked** when displayed on the interface and will **not be stored for security reasons**. The **payment transaction details** will be **encrypted** before invoking core service to **process and persist** payment transaction in Payment data store and payment status response will be presented to the user.

3.3 Application architecture

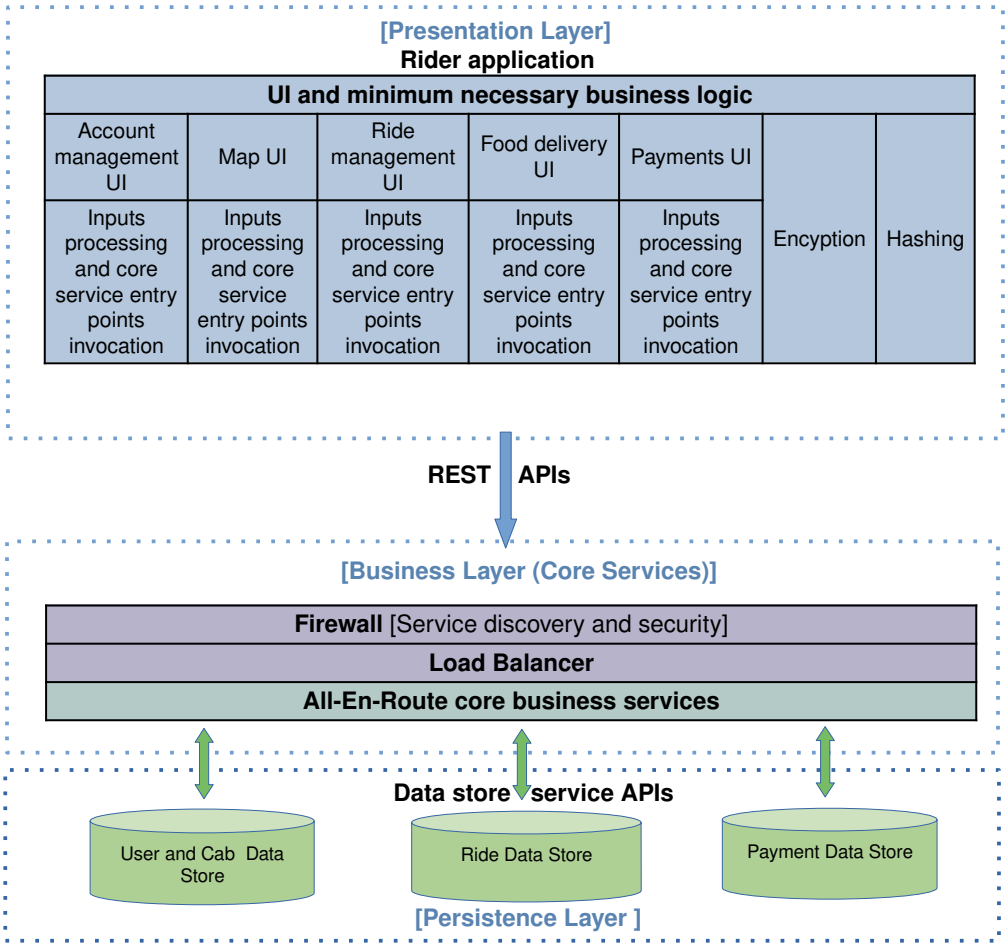

Fig 2: Architecture for Rider Application

Rider application will be available as mobile application for passengers and web application for kiosk operators. It will primarily provide user interface for the functionality described in section 3.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route.
The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS. The web application will run in browser (Chrome, Internet Explorer and Safari) on laptop.

4. Driver Application

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

4.1 Application function level forward traceability

Functionality [Ref: Section 2 in Functional Architecture document]	**All-En-Route Driver Application [Ref: Section 4 in this document]**
2. Functionality for Cab providers	4. Driver Application
2.1 Account management	4.2.1 Account management interface
2.1.1 Onboarding	4.2.1.1 Onboarding
2.1.2 Login and logout	4.2.1.2 Login and logout
2.1.3 Profile Management	4.2.1.3 Profile
2.2 Mapping services	4.2.2 Map interface
2.3 Ride management	4.2.3 Ride management interface
2.3.1 Ride request assignment	4.2.3.1 Ride assignment
2.3.2 Ride processing	4.2.3.2 Ride processing
2.3.3 Ride cancellation	4.2.3.3 Ride cancellation
2.4 Payments management	4.2.4 Payments interface

4.2 Application modules and functional flows

The application will have following **modules**:

- Account management interface
- Map interface
- Ride management interface
- Payments interface

4.2.1 Account management interface

This module will provide interface for **onboarding, authentication and profile management** functionality described in “Section 2.1. of Functional Architecture” document.

4.2.1.1 Onboarding

Onbording interface option will be available for driver (cab provider) on application startup. As described in “Section 2.1.1 of Functional Architecture” document, the driver will use registration UI screen to enter registration details and upload documents along with cab details. Documents will be have to be uploaded in form of **JPG** or **PDF**.
Mandatory input checks and **standard input format validations** will be performed on the information entered by driver. Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security. The interface will then invoke core service entry point to persist the driver and cab details, upload documents to User and Cab data store and start the verification process. Phone number and e-mail will be verified using OTP and e-mail verification link. Uploaded documents will be verified offline for authenticity within **2-3 working days**. On successful verification, cab drivers will be notified via phone and e-mail and they will need to **agree on a contract specifying all the terms and conditions**. On successful verification and contract signing, cab and driver details persisted will be marked active and driver will be notified of the same.

4.2.1.2 Login and logout

Successfully onboarded cab providers will use login UI screen to log into the application. UserID and password inputs will be provided by user on login screen. The **password** will be **one-way-hashed**. The user entered credentials will be verified against persisted credentials in the User data store using core service entry point. On successful verification, user will be logged into the application and can start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt login again with maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

4.2.1.3 Profile

Once successfully logged in, user can choose to update the profile information. Once user confirms updates to the profile, **PII** information in the profile details will be **encrypted**. The user entered profile updates will be persisted in the User data store using core service entry point and status will be communicated to the user.

4.2.2 Map interface

This module will provide interface for viewing assigned ride source and destination along with driving route and directions as described in “Section 2.2 of Functional Architecture” document.
The user will see a map view on the screen **showing the source and destination locations of the ride**, the **route to take** and the estimated **time** of arrival (ETA) of destination. **On ride start, driving diections** will be provided in real-time. The information to populate the UI will be obtained using core service entry points.

4.2.3 Ride management interface

This module will provide interface for **ride assignment, ride processing and ride cancellation** functionality as described in “Section 2.3 of Functional Architecture” document.

4.2.3.1 Ride assignment

The **cab providers will see their matched rides in the UI.** The matching will take place in core services as per the business rules and requirements, as described in the “Section 2.3.1 of Functional Architecture” document. The cab provider will be able to **select and accept** one from available ride requests. The service platform will then assign the ride request to the cab provider using criteria as described in the “Section 2.3.1 of Functional Architecture” document. The cab provider will be then notified of ride assignment. The cab will be marked busy from the beginning of the ride until 5 minutes before the end time.
The matched ride data will be obtained by invoking core service entry point. Once a ride is accepted by driver, ride request will be updated in Ride data store.

4.2.3.2 Ride processing

To start a ride, the driver will get a UI screen to enter an OTP provided by the passenger. The OTP data will be validated by invoking core service entry point. On successful OTP validation, the ride will begin. Cab location tracking (source, destination, route, current location and ETA) will be available to the rider and the driver till the ride ends. On reaching the destination, the driver will use an option in the UI to finish the ride, after which the payment screen will be available to passenger.

4.2.3.3 Ride cancellation

The driver will have an option to cancel accepted and assigned ride as per the rules described in “Section 2.3.3 of Functional Architecture” document. The ride cancellation will be updated in ride request status in the Ride data store by invoking core service entry point.

4.2.4 Payments interface

The payments for driver (cab providers) will be settled on daily basis using daily batch jobs in the system. This module will provide interface for **ride payment** functionality as described in “Section 2.4 of Functional Architecture” document. Payments to drivers (cab providers) will be **settled** by the service platform **on a daily basis**. The drivers (cab providers) will receive consolidated statement for the rides completed and penalties, if any, for the day, on their registered e-mail. The same can also be seen in the interface of the application. The payment records will be **encryted** while in-transit and at rest and will only be accessible on entering the password. The payment records will be obtained from the Payment data store by invoking core service entry point.

4.3 Application architecture

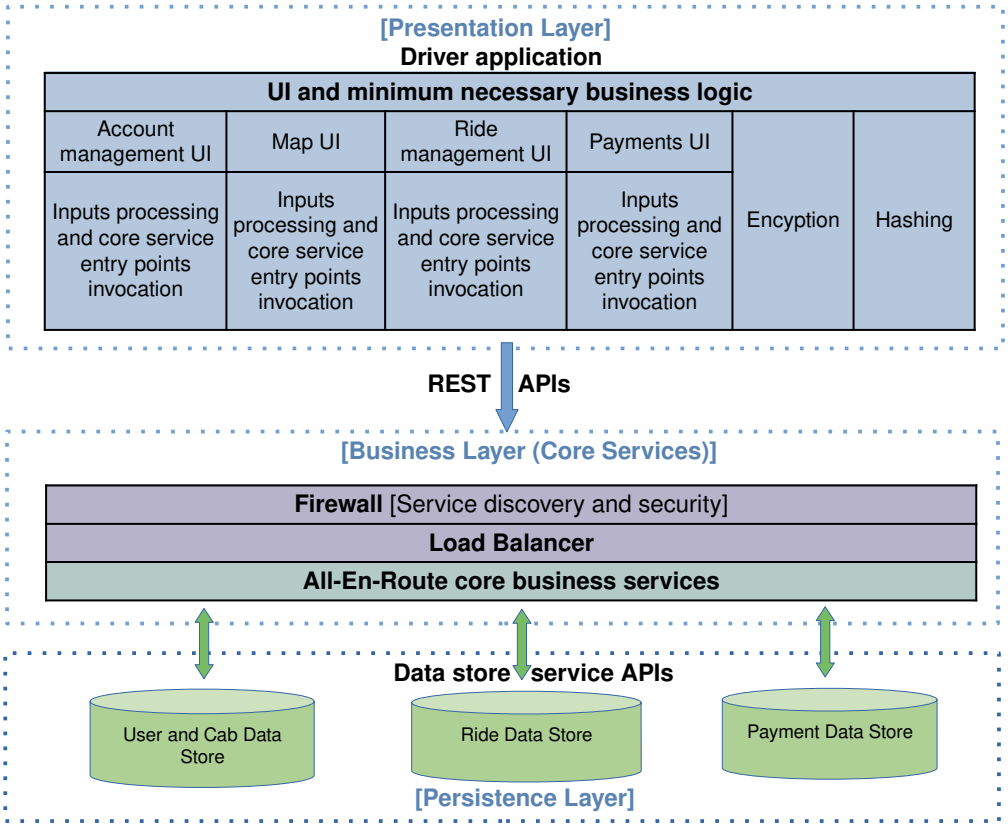

Fig 3: Architecture for Driver Application

Driver application will be available as mobile application for drivers. It will primarily provide user interface for the functionality described in section 4.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route.
The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS.

5. Operations Application

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

5.1 Application function level forward traceability

Functionality [Ref: Sections 3, 4, 5 and 6 in Functional architecture document]	**All-En-Route Operations Application [Ref: Section 5 in this document]**
4. Functionality for system administrator	5. Operation Application
3,4,5,6 Standard login and logout	5.2.1 Authentication interface
4.1 User account management	5.2.2 Account management interface
4.2 Service platform installation management	5.2.3 Service platform management interface
5. Functionality for customer support personnel	5. Operation Application
5.1 Customer issue processing	5.2.4 Customer issue interface
5.2 Support knowledge base search	5.2.5 Knowledge base interface
6. Functionality for technical support personnel	5. Operation Application
6.1 Customer issue processing	5.2.6 Technical issue interface
6.2 Support knowledge base update	5.2.7 Knowledge base management interface
3. Functionality for cab aggregator (business owner)	5. Operation Application
3.1 Basic charts	5.2.8 Basic charts interface
3.2 Analytics charts	5.2.9 Analytics charts interface
3.3 Contracts and compliance information	5.2.10 Dashboard interface

5.2 Application modules and functional flows

The application will have following **modules:**

- Authentication interface
- Account management interface
- Service platform management interface
- Customer issue interface
- Knowledge base interface
- Technical issue interface
- Knowledge base management interface
- Basic charts interface
- Analytics charts interface
- Dashboard interface

5.2.1 Authentication interface

This module will provide **user authentication** and **authorization** functionality using role based access control **(RBAC)**. The **roles** - System administrator, Customer support personnel, Technical support personnel and Business owner will be **preassigned** to the users. User will need to change the password on first login. User then will enter **user ID** and **password** to log into the application. For **privacy** reasons, the password will be **masked** while being entered. It will then **invoke core service entry point to verify the credentials (with password one-way-hashed for security)** and get the preassigned role. On successful verification, user will be logged into the application with preassigned role with RBAC enforced, permitting only the functionalities authorized for the role. On failure, user will be provided with appropriate error messages and allowed to attempt login again with maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

5.2.2 Account management interface

The system administrator and business owner accounts will be preconfigured for the system. The customer support personnel, technical support personnel and kiosk operator accounts will be created by the system administrator as needed.
This module will be only available to **system administrator**. It will provide the administrator with an UI interface to **Create**, **View**, **Update** and **Delete** (CRUD) customer support personnel, technical support personnel and kiosk operator accounts for the service platform. While updating any account, Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security. Administrator can also **block** the cab providers who fail to comply with automated periodic regulatory checks. The account registrations and profile updates will be persisted in the User data store using core service entry point.

5.2.3 Service platform management interface

This module will allow **system administrator** to **install, configure, upgrade, uninstall All-En-Route software** on the initial hardware and **apply security patches** for the service platform. It will provide separate UI for **service platform health and resource capacity monitoring**. Administrator will also be able to **manage and configure batch jobs for** driver (cab provider) and food-delivery service partner daily **payment settlements** and automated periodic regulatory **compliance checks** of the drivers and cabs. The install, configure, upgrade, uninstall and patch processes will use the native mechanism available in the deployment environment. The batch jobs will use core service entry points for payments processing. The compliance checks will start offline verification process for verifications due as per regulations. The verification process statuses will be uploaded to the system and batch jobs will produce list of drivers and/or cabs that will need to be deactivated due to non-compliance.

5.2.4 Customer issue interface

This module is for **customer support personnel**. It will show the customer raised issues sorted with severity (high, medium or low) level as mentioned by the issuer. The personnel will be able to **view** the assigned issues and **resolve** or **forward** it to the technical support. If the issue is resolved, customer will be notified accordingly. The updates will be persisted in the Operations data store using the core service entry point.

5.2.5 Knowledge base interface

This module is for **customer support personnel**. It will provide an UI based interface to **search** for the customer issue from the Knowledge Base (KB). If the resolution is present in KB, the personnel will apply the same solution and **resolve** the issue or else forward it to technical support. It will invoke the core service entry point to search the knowledge base in the Operations data store.

5.2.6 Technical issue interface

This module is for **technical support personnel**. It will show the issues forwarded by the customer support in severity order. The personnel will be able to **resolve** the issue and update the issue status. The updates will be persisted in the Operations data store using the core service entry point.

5.2.7 Knowledge base management interface

This module is for **technical support personnel**. It invokes the core service entry point to manage the Knowledge Base. The personnel can add, remove or update the resolved technical issue to the knowledge base for future reference by the customer support.

5.2.8 Basic charts interface

This module is for the **business owner**. The basic charts functionality will present user with predefined charts and allow user to select the dimensions, metrics and time granularity from prepopulated lists for the chart. It will invoke core service entry point to obtain the chart data for selected attributes from Analytics data store.

5.2.9 Analytics charts interface

This module is for the **business owner**. This will allow the user to **configure comparison charts** and **trends charts and** allow user to **explore basic charts on second dimension**. It will invoke the core service entry point to persist the configured chart definitions and obtain the chart data from Analytics data store. The user will be allowed to delete any charts.

5.2.10 Dashboard interface

This module is for the **business owner**. The dashboard will provide information regarding regulatory compliance statuses, business licenses, partner service subscription or contract fees and renewal due dates.

5.3 Application Architecture

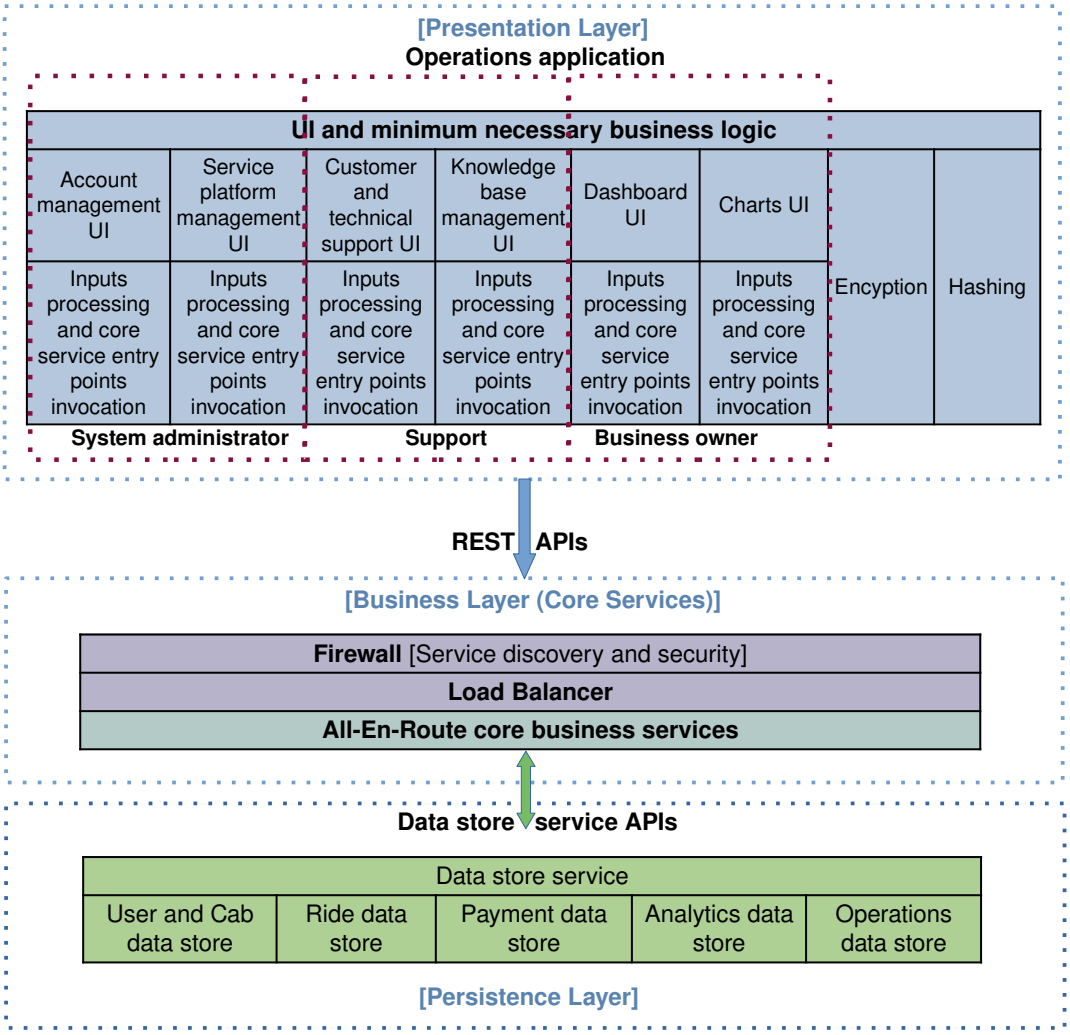

Fig 4: Architecture for Operations Application

Operations application will be available as web application for system administrator, customer support personnel, technical support personnel and as web application as well as mobile application for business owner. It will primarily provide user interface for the functionality described in section 5.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route. The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS. The web application will run in browser (Chrome, Internet Explorer and Safari) on laptop.

6. All-En-Route core and platform services

Based on characteristics of various architectural styles for loosely coupled highly cohesive modularization with anytime anywhere availability, responsiveness, deployability and maintainability requirements, **All-En-Route core business logic will use service oriented architecture** and will be manifested as microservices. All the service entrypoints will use common service entry point for logging.

All-En-Route business logic will be available in following **core services**:

- UserManagement
- MappingService
- RideManagement
- PaymentProcessing
- Food-deliveryProcessing
- AnalyticsService

Following are **common services** used by core services:

- DataStoreService
- ValidationService
- AuditService

6.1 User management service (UserManagement)

This service will provide user account management functionality for registration, authentication, and profile management.
It will have following entrypoints:

- GenerateOTP
- ValidateOTP
- GenerateEmailVerificationLink
- ValidateEmailVerificationLink
- RegisterUser
- AuthenticateUser
- UpdateProfile

6.1.1 GenerateOTP

For given user ID and phone number inputs, it will generate OTP for given user ID using ValidationService:GenerateOTP and send it to user phone number using ValidationService:SendOTP. The OTP will be stored in User data store marked for deletion after its expiry.

6.1.2 ValidateOTP

For given user ID and phone number inputs and OTP, it will validate OTP using ValidationService:ValidateOTP for match and expiry. It will remove OTP from User data store once processed, using ValidationService:DeleteOTP.

6.1.3 GenerateEmailVerificationLink

For given user ID and e-mail address inputs, using ValidationService:GenerateEmailVerificationLink it will generate e-mail verification link for given user ID and send the link to user e-mail address using ValidationService:SendEmailVerificationLink. The e-mail verification link will be stored in User data store marked for deletion after its expiry.

6.1.4 ValidateEmailVerificationLink

For given user ID and e-mail address inputs and e-mail verification link, using ValidationService:ValidateEmailLink it will validate e-mail verification link for match and expiry. Once processed, it will remove e-mail verification link using ValidationService:DeleteEmailLink from User data store.

6.1.5 RegisterUser

It will persist input user registration details in User data store using DataStoreService:CreateOrUpdateDataRecord.

6.1.6 AuthenticateUser

It will validate input user credentials against user credentials available in User data store using VerificationService:ValidateCredentials, it will return the status and preassigned role of the validated user.

6.1.7 UpdateProfile

It will persist input profile registration details in User data store using DataStoreService:CreateOrUpdateDataRecord.

6.2 Mapping service (MappingService)

This service will invoke predetermined external geocoding and routing service provider interfaces to get the required functionality. Using the external mapping service provider interfaces, it will provide for **location latitude and longitude (lat-long), source to destination routes and cab location tracking** functionality. It will have following **entrypoints**:

- GetMapViewForLocation
- GetMapViewForRoute
- StartMapViewUpdatesForCabLocation
- StopMapViewUpdatesForCabLocation

6.2.1 GetMapViewForLocation

For given textual location input, GetMapViewForLocation will **call the predetermined external geocoding and mapping service provider interface to get** the **lat-long and map view information** for the (Rider and Driver) application to render the map view with location marker.
[The external geocoding service provider to use will be predetermined after evaluating options (such as Google Maps Geocoding API or TomTom Search API) during design phase.]

6.2.2 GetMapViewForRoute

For given source and destination location inputs, GetMapViewForRoute will **call the predetermined external routing and mapping service provider interface to get the routing and map view information** for the (Rider and Driver) application to render the map view with route.
[The external routing service provider to use will be predetermined after evaluating options (such as Google Maps Directions API or TomTom Routing API) during design phase.]

6.2.3 StartMapViewUpdatesForCabLocation (Cab Location Tracking)

For given ride ID input for current ride**, long polling** will be setup for Rider and Driver applications **to receive initial map view information with source, destination, route for the ride and continuous map view updates for cab location** based on drivers smartphone GPS coordinate updates received periodically.

6.2.4 StopMapViewUpdatesForCabLocation (Cab Location Tracking)

On current ride finish for given cab ID, **long polling setup** for Rider and Driver applications to receive map view updates for cab location **will be taken down**.

6.3 Ride management service

This service will provide for **route, tariff and cab availability for given ride, scheduling, assignment and cancellation of rides** functionality. It will have following **entrypoints**:

- GetRideOptions
- CreateAndAssignRide
- GetAssignedCabDetails
- AssignRide
- StartRide
- EndRide
- CancelRide

6.3.1 GetRideOptions

This entrypoint provides **map view with source, destination locations, route, distance, ETA, tariff (by cab types) and cabs available in the vicinity of source location** for ride information. For given source and destination inputs, GetRideOptions will call **GetMapViewForLocation and GetMapViewForRoute entrypoints of the MapService** to **get map view with location lat-longs** and **route** information. It will **use** the **route distance and ETA information to compute tariffs for various cab types**. Based on the continuous **cabs location** tracking (GPS Coordinates) **and Geospatial Indexing**, cab locations in the vicinity will be obtained **for reporting** on map view interface.

6.3.2 CreateAndAssignRide

For given ride (Now), CreateRide will **create ride request**, **persist** it in ride data store and will **assign** using the logic described in “Section 2.3.1.1 of Functional Architecture” document for regular ride assignment and “Section 2.3.2.2 of Functional Architecture” document for long-distance ride assignment. It will return the ride ID and status of processing once the ride request is created and ride assignment will continue asynchronously. The status of assigned ride will be updated in Ride data store once done.
For given ride with schedule time (!= Now), CreateRide will **create ride request**, **persist** it in ride data store. Ride request will be associated **with timer** that will go off at [ride start time – predefined buffer]. The predefined buffer will be 30 minutes for regular rides and 2 hours for long distance rides. When the timer goes off, **ride assignment** will **begin**. It will return the ride ID and status of processing once the ride request is created and ride assignment will continue asynchronously. The status of assigned ride will be updated in Ride data store once done.

6.3.3 GetAssignedCabDetails

When invoked with a ride ID, it will return **current status of ride assignment and cab details, driver details and ride OTP if assignment is done** from Ride data store.

6.3.4 AssignRide

For given ride ID and driver ID inputs, AssignRide will update the status of the ride to **ASSIGNED** in the Ride data store and ride request will be updated with driver ID and cab ID.

6.3.5 StartRide

For given ride ID marked started with OTP, StartRide will match the OTP in the persisted ride request and update the status of the ride request to **STARTED** along with start time in the Ride data store.

6.3.6 EndRide

For given ride ID marked finished by user, EndRide will update the status of the ride to **FINISHED** along with end time in the Ride data store.

6.3.7 CancelRide

For given ride ID marked cancelled by user, CancelRide will update the status of the ride to **CANCELLED** along with cancellation time in the Ride data store.

6.4 Payment processing service

This service will process rides (food-delivery and penalties) payments for rides marked FINISHED or CANCELLED, daily driver (cab provider) payments and daily food-delivery service partner payments. It will call payment gateway service interfaces. It will have following **entrypoints**:

- GetRideDues
- ProcessPassengerPayment
- ProcessDriverPayment
- ProcessFoodDeliveryPartnerPayments

6.4.1 GetRideDues

When invoked with ride ID, GetRideDues will compute the ride payment dues based on ride tariff, food bills of the ride, cancellation charges for current ride if applicable (based on cancellation status and start and cancellation time of the ride) and passenger penalties pending if any.

6.4.2 ProcessPassengerPayment

When invoked with payment method type and payment method details, it will call payment gateway service interface for given payment method with given payment method details. The payment transactions will be stored in Payment data store.

6.4.3 ProcessDriverPayment

When invoked with list of driver (cab provider) payment transactions for the day, it will fetch payment method type and details for each driver for each transaction. It will then call payment gateway service interface for the payment method with payment method details to make the payments. All the payment transactions will be stored in Payment data store.

6.4.4 ProcessFoodDeliveryPartnerPayments

When invoked with list of food-delivery service partner(s) payment transactions for the day, it will fetch payment method type and details for the food-delivery service partner for each transaction. It will then call payment gateway service interface for the payment method with payment method details. All the payment transactions will be stored in Payment data store.

6.5 Food-delivery processing service

This service will provide food stops listing, food menu listing, food-deliver ordering and status interface using respective interfaces of food-delivery partner service and present that information in the Rider application UI. It will have following **entrypoints**:

- ListFoodStops
- ListFoodOptions
- OrderFoodDelivery

6.5.1 ListFoodStops

When invoked with route details, it will compute food stops available on given route as per the logic described in “Section 1.4.1 in Functional Architecture” document. It will persist the food stops in the ride request in Ride data store.

6.5.2 ListFoodOptions

When invoked with food stop input, it will query the food-delivery partner service(s) for food options available for given food stop.

6.5.3 OrderFoodDelivery

When invoked with food stop and food options selected for ordering inputs, it will place the food delivery order for selected food options using food-delivery partner service(s) interfaces and persist the food-delivery order in ride request in Ride data store.

6.6 Analytics service

This service will provide functionality for predefined and configurable charts, trends, comparisons insights for business owners. It will have following **entrypoints**:

- GetCharts
- ConfigureChart
- DeleteChart

6.6.1 GetCharts

When invoked, it will **return currently predefined basic and preconfigured analytics charts data** from Analytics data store.

6.6.2 ConfigureChart

When invoked with **type of chart, dimensions, metrics and time granularity**, it will **generate analytics chart query that will run as per given time granularity** and **persist** it in Analytics data store **as preconfigured chart**.

6.6.3 DeleteChart

When invoked with chart ID, it will **delete** that **predefined or preconfigured chart definition** from Analytics Data Store.

6.7 Data store service

This platform service will provide data create, read, update and delete (CRUD) operations to all the core services for specified data stores. It will use standard opensource NoSQL database that supports required data models. The NoSQL database to use will be predetermined after evaluating options (such as Cassandra or MongoDB) during design phase. It will have following **entrypoints**:

- CreateOrUpdateDataRecord
- GetDataRecord
- DeleteDataRecord

6.7.1 CreateOrUpdateDataRecord

When invoked with data record and data store, it will create or update the data record in the given data store and return the status.

6.7.2 GetDataRecord

When invoked with query attributes and data store, it will retrieve the data record based on given query attributes from the given data store and return the status and data record if found.

6.7.2 DeleteDataRecord

When invoked with attributes and data store, it will delete the data record matching the given attributes in the given data store and return the status.

6.8 ValidationService

This common service will provide OTP generation, validation, e-mail link generation and validation functions. It will have following **entrypoints:**

- GenerateOTP
- SendOTP
- ValidateOTP
- DeleteOTP
- GenarateEmailVerificationLink
- SendEmailVerificationLink
- ValidateEmailVerificationLink
- DeleteEmailVerificationLink

6.8.1 GenerateOTP

GenerateOTP function will generate an OTP for given user ID or ride ID.

6.8.2 SendOTP

SendOTP will sms the OTP to the user associated with the user ID or ride ID.

6.8.3 ValidateOTP

ValidateOTP will validate OTP for given user ID or ride ID against the persisted OTP in user record in User data store or ride request in Ride data store.

6.8.4 DeleteOTP

DeleteOTP will delete the OTP after processing or expiry.

6.8.5 GenerateEmailVerificationLink

GenerateEmailVerificationLink function will generate an EmailVerificationLink for given user ID.

6.8.6 SendEmailVerificationLink

SendEmailVerificationLink will email the EmailVerificationLink to the user associated with the user ID.

6.8.7 ValidateEmailVerificationLink

ValidateEmailVerificationLink will validate EmailVerificationLink for given user ID against the persisted EmailVerificationLink in user record in User data store.

6.8.8 DeleteEmailVerificationLink

DeleteEmailVerificationLink will delete the EmailVerificationLink after processing or expiry.

6.9 AuditService

This common service will provide WriteLog entrypoint to write log records to Operations data store.

7. Integration Architecture

7.1 Internal and External integrations

Internal integration refers to integration across multiple applications of the system and external integration refers to integration with external systems.

Internally, the All-En-Route user applications will interact with the All-En-Route core services using REST API entry points. The services will persist and retrieve data using data store services.

External services used by All-En-Route will include external mapping services, external food-delivery partner services and payment gateway services. All-En-Route core services will invoke the interfaces provided by these external services.

Table below depicts the integrations among the applications and services in the All-En-Route system. Invocations by user applications will be inbound to the invoked service which will provide the respective entry point. Invocations of external service (payment gateway, mapping and food delivery partner services) entry points by All-En-Route services will be outbound and will be provided by external services.

ID	Invoking Application	Invoked Service	Integration Description	Inbound or Outbound	Data integration or functional integration
7.2.1	Rider Application	UserManagement	Passenger details verification	Inbound	Functional integration
7.2.2	Rider Application	UserManagement	Passenger registration	Inbound	Functional integration
7.2.3	Rider Application	UserManagement	Passenger login and logout	Inbound	Functional integration
7.2.4	Rider Application	UserManagement	Passenger Profile Update	Inbound	Functional integration
7.2.5	Rider Application	RideManagement	Passenger map views with location and routes	Inbound	Functional integration
7.2.6	Rider Application	MappingService	Cab location tracking	Inbound	Functional integration

7.2.7	Rider Application	RideManagement	Ride request creation and assignment	Inbound	Functional integration
7.2.8	Rider Application	RideManagement	Cab assignment notification	Inbound	Functional integration
7.2.9	Rider Application	RideManagement	Ride cancellation by passenger	Inbound	Functional integration
7.2.10	Rider Application	PaymentProcessing	Ride payment	Inbound	Functional integration
7.2.11	Rider Application	Food-deliveryProcessing	Food stops listing	Inbound	Functional integration
7.2.12	Rider Application	Food-deliveryProcessing	Food options listing	Inbound	Functional integration
7.2.13	Rider Application	Food-deliveryProcessing	Food order	Inbound	Functional integration
7.2.14	Driver Application	UserManagement	Driver details verification	Inbound	Functional integration
7.2.15	Driver Application	UserManagement	Driver onboarding	Inbound	Functional integration
7.2.16	Driver Application	UserManagement	Driver login and logout	Inbound	Functional integration
7.2.17	Driver Application	UserManagement	Driver Profile Update	Inbound	Functional integration
7.2.18	Driver Application	MappingService	Driver map views with location, routes, location tracking and directions	Inbound	Functional integration
7.2.19	Driver Application	RideManagement	Ride assignment	Inbound	Functional integration
7.2.20	Driver Application	RideManagement	Ride processing	Inbound	Functional integration
7.2.21	Driver Application	RideManagement	Ride cancellation by driver	Inbound	Functional integration
7.2.22	Driver Application	PaymentProcessing	Daily Payment Settlement	Inbound	Functional integration
7.2.23	Operations Application	UserManagement	Authentication	Inbound	Functional integration
7.2.24	Operations Application	UserManagement	Account management	Inbound	Functional integration
7.2.25	Operations Application	DataStoreService	Customer issues	Inbound	Functional integration
7.2.26	Operations Application	DataStoreService	Knowledge base search	Inbound	Functional integration
7.2.27	Operations Application	DataStoreService	Technical issues	Inbound	Functional integration
7.2.28	Operations Application	DataStoreService	Knowledge base update	Inbound	Functional integration
7.2.29	Operations Application	AnalyticsService	Basic charts	Inbound	Functional integration
7.2.30	Operations Application	AnalyticsService	Analytics charts	Inbound	Functional integration
7.2.31	Operations Application	AnalyticsService	Analytics charts	Inbound	Functional integration
7.2.32	Operations Application	DataStore	Dashboard for business owner	Inbound	Functional integration
7.2.33	MappingService	Geocoding, External Mapping service	Location map view	Outbound	Functional integration
7.2.34	MappingService	Routing, External Mapping service	Route map view	Outbound	Functional integration
7.2.35	RideManagement	Geospatial Indexing, External Mapping service	Cabs in vicinity and map view	Outbound	Functional integration
7.2.36	Food-deliveryProcessing	Food-delivery partner service	Food options	Outbound	Functional integration
7.2.37	Food-deliveryProcessing	Food-delivery partner service	Food order	Outbound	Functional integration
7.2.38	PaymentProcessing	PaymentGateway	Payment processing	Outbound	Functional integration
7.2.39	All services	DataStoreService	Data record CRUD	Inbound	Functional integration
7.2.40	All services	AuditService	Logging	Inbound	Functional integration
7.2.41	All applications	ValidationService	Phone and e-mail verification	Inbound	Functional integration

7.2 Integration details

7.2.1 Passenger details verification

On successful input checks of the information entered by passenger, the **Signup** functionality in **Rider application** will invoke **GenerateOTP entrypoint of UserManagement service** to generate and send the OTP and will invoke **ValidateOTP to verify** the **phone number using OTP.** It will invoke **GenerateEmailVerificationLink entrypoint of UserManagement service** to generate and send e-mail verification link and will invoke **ValidateEmailVerificationLink to verify** the **e-mail** address **using e-mail verification link**.

7.2.2 Passenger registration

On successful verification of user details and user confirmation for registration, the **Signup** functionality in **Rider application** will invoke **RegisterUser entrypoint of UserManagement service** to persist passenger details in User data store. The return status will be communicated to the user.

7.2.3 Passenger login and logout

After encrypting user (passenger) credentials, the **AuthenticateUser entrypoint of UserManagement service** will be **invoked** to validate the credentials against the credentials information available in the User data store and status will be communicated to the user.

7.2.4 Passenger profile update

After encrypting PII information in the profile details, the **UpdateProfile entrypoint of UserManagement service** will be invoked to persist the profile updates and status will be communicated to the user.

7.2.5 Passenger map views with location, routes, ride options

The map view inteface of Rider application will invoke **GetRideOptions entrypoint of RideManagement service** to get the required map view details with source, destination, route, ride options with tariff and ETA information for rendering the map view for the passenger.

7.2.6 Cab location tracking

Once the cab is assigned, the map view interface of Rider application will invoke **StartMapViewUpdatesForCabLocation entrypoint of MappingService service** with the ride ID for cab location tracking using long polling setup to get initial map view information with source, destination, route and continuous map view updates for current cab location. The cab location updates will be available throughout the ride.

7.2.7 Ride request creation and assignment

Once the rider selects the ride option and confirms ride, the Rider application will invoke **CreateAndAssignRide entrypoint of RideManagement service.** The service entrypoint will return the ride ID and status of the ride creation. The assignment will happen asynchronously and will be push notified or polled for status.

7.2.8 Cab assignment notification

The cab assignment screen of Rider application will be refreshed periodically for assigned cab and driver details. The updates are fetched periodically using push notifications from RideManagement service or by calling **GetAssignedCabDetails entrypoint of RideManagement service.**

7.2.9 Ride cancellation by passenger

On cancellation by passenger, Rider application invokes **CancelRide entrypoint of RideManagement service.** The service entrypoint will update the status of the ride request as cancelled in the Ride data store.

7.2.10 Ride payment

The payment interface will be activated once the ride is marked finished. The bill information will be either fetched via push notifications or by periodically invoking **GetRideDues entrypoint of PaymentService**. For payment processing, once the payment method is selected, details are entered (masked) and encrypted, Rider application will invoke **ProcessPassengerPayment entrypoint of PaymentService.** The payment transaction will be processed by the service entrypoint and persisted in the Payment data store.

7.2.11 Food stops listing

Once the ride is confirmed, Rider application will invoke **ListFoodStops entrypoint of Food-deliveryProcessing service** to get the food stops to be presented to the passenger.

7.2.12 Food options listing

Once the passenger selects food stop, Rider application will invoke **ListFoodOptions entrypoint of Food-deliveryProcessing service** to get the food options at selected food stop to be presented to the passenger.

7.2.13 Food order

Once the passenger selects food from food options to order, the Rider application will call **OrderFoodDelivery entrypoint of Food-deliveryProcessing service**. The service entrypoint will call outbound interface of food-delivery partner service to place the order and get the confirmation status. It will then respond back with status after updating the ride request with the associated food-order in Ride data store.

7.2.14 Driver details verification

On successful input checks of the information entered by driver, the Onboarding functionality in **Driver application** will invoke **GenerateOTP entrypoint of UserManagement service** to generate and send the OTP and will invoke **ValidateOTP to verify** the **phone number using OTP.** It will invoke **GenerateEmailVerificationLink entrypoint of UserManagement service** to generate and send e-mail verification link and will invoke **ValidateEmailVerificationLink to verify** the **e-mail** address **using e-mail verification link**.

7.2.15 Driver onboarding

On successful verification of driver details and start of onboarding process, the **Onboarding** functionality in **Driver application** will invoke **RegisterUser entrypoint of UserManagement service** to persist driver and cab details and documents provided in User data store. The return status will be communicated to the user. After successful offline verification of documents, activation of the driver and cab will be done by system administrator via daily automated batch job of activating verified drivers and cabs and sending SMS and e-mail notifications to the respective drivers (cab providers).

7.2.16 Driver login and logout

After encrypting user (driver) credentials, the **AuthenticateUser entrypoint of UserManagement service** will be **invoked** to validate the credentials against the credentials information available in the User data store and status will be communicated to the user.

7.2.17 Driver Profile Update

After encrypting PII information in the profile details, the **UpdateProfile entrypoint of UserManagement service** will be invoked to persist the profile updates and status will be communicated to the user.

7.2.18 Driver map views with location and routes

Once the driver starts the ride using Driver application UI, it will invoke **StartMapViewUpdatesForCabLocation entrypoint of MappingService serivce** with the ride ID for cab location tracking using long polling setup to get initial map view information with source, destination, route and continuous map view updates for current cab location. The cab location updates will be available throughout the ride.

7.2.19 Ride assignment

The driver will be notified with available matched ride requests list via push notification from RideManagement service. Once the driver accepts one of the ride from the list, Driver application will invoke **AssignRide entrypoint of RideManagement service.** The ride request will be updated by the service entrypoint for assigned driver and cab in Ride data store.

7.2.20 Ride processing

When the driver starts the ride by entering OTP provided by the passenger, the Driver application will invoke **StartRide entrypoint of RideManagement service.** The service entrypoint will update the status of ride request after matching the OTP provided.
When the driver marks the ride finished using Driver application UI, the Driver application will invoke **EndRide entrypoint of RideManagement service.** The service entrypoint will update the status of ride request.

7.2.21 Ride cancellation by driver

On cancellation by driver, Driver application invokes **CancelRide entrypoint of RideManagement service.** The service entrypoint will update the status of the ride request as cancelled in the Ride data store.

7.2.22 Daily Payment Settlement

System will run configured daily automated batch job to settle drivers (cab providers) payments. It will prepare the list of driver (cab providers) payment transactions and call **ProcessDriverPayment entrypoint of PaymentService** with this list. The service entrypoint will execute the payment transactions to settle the daily driver (cab provider) dues.

7.2.23 Authentication

Operations application will invoke **AuthenticateUser entrypoint of UserManagement service** to validate the credentials against the credntials information available in the User data store and status and preassigned role of the user will be returned.

7.2.24 Account management

For user profiles created by system administrator in Operations application for support staff and kiosk operators, Operations application will invoke **RegisterUser entrypoint of UserManagement service** to register and persist user details in User data store. The return status will be communicated back.

7.2.25 Customer issues

For populating customer issues assigned for logged in customer support personnel, Operations application will **invoke GetDataRecord entrypoint of the DataStore service** to query assigned issues data records from Operations data store for given user ID.

7.2.26 Knowledge base search

For issue resolution, customer support personnel will use Operations application UI to **invoke GetDataRecord entrypoint of the DataStore service** to query resolution data records from Operations data store for specific issue keywords.

7.2.27 Technical issues

For populating technical issues forwarded by customer support personnel and assigned to logged in technical support personnel, Operations application will **invoke GetDataRecord entrypoint of the DataStore service** to query assigned issues data records from Operations data store for given user ID.

7.2.28 Knowledge base update

For issues resolved, technical support personnel will use Operations application UI to **invoke CreateOrUpdateDataRecord entrypoint of the DataStore service** to create or update resolution data records in Operations data store for specific issue keywords with associated obtained resolution.

7.2.29 Basic charts

On selection of predefined chart dimension, metric and time granularity by business owner, Operations application will **invoke GetChart entrypoint of the Analytics service** to fetch the chart data to render by running the chart specifc query in Analytics data store.

7.2.30 Analytics charts

When business owner **configures and save an analytics chart definition** with specifc type (comparison, trend, second dimension), dimension, metric and time granularity, Operations application will **invoke ConfigureChart entrypoint of the Analytics service** to **persist the chart definition** in the Analytics data store and then **invoke GetChart entrypoint of the Analytics service to fetch the chart data** to render by running the chart specifc query in Analytics data store.

7.2.31 Delete chart

When the business owner **deletes** a **chart**, **DeleteChart entrypoint of the Analytics service** will be invoked to delete the chart definition from Analytics data store.

7.2.32 Dashboard for Business Owner

On login, business owner will be presented with Dashboard. The information on regulatory compliance statuses, business licenses, partner service subscription or contract fees and renewal due dates will be displayed on dashboard which will be fetched using **GetDataRecord entrypoint of the DataStore service** with predefined query attributes.

7.2.33 Location information

MappingService will invoke predetermined **Geocoding APIs to get Location lat-long coordinates for given textual location address** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with location markers**.

7.2.34 Route information

MappingService will invoke predetermined **Routing APIs to get route information for given source and destination locations** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with the route information.**

7.2.35 Cabs in vicinity and map view

RideManagement service entrypoint GetRideOptions will use driver (cab provider) smartphone GPS coordinates and invoke predetermined **Geospatial Indexing APIs to determine cabs in the vicinity of source location of the ride** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with the cabs in vicinity information.**

7.2.36 Food options

Food-deliveryProcessing service will **use Food-delivery partner service API to get the food options available at specific food stop if any**.

7.2.37 Food order

Food-deliveryProcessing service will **use Food-delivery partner service API to order food at specific food stop**.

7.2.38 Payment processing

PaymentProcessing service will **use Payment gateway** APIs (such as PayPal) **to execute the payment transactions**.

7.2.39 Data record CRUD

All services will use **DataStoreService entrypoints CreateOrUpdateDataRecord, GetDataRecord and DeleteDataRecord** for the data record CRUD operations.

7.2.40 Logging

All services will use **AuditService WriteLog entrypoint** for writing log records to Operations data store.

7.2.41 Phone and e-mail verification

All applications will use **ValidateOTP and ValidateEmailVerificationLink entrypoints of ValidationService** for validating the phone and e-mail addresses.

8. References

[1] Service-Oriented Architecture: Concepts, Technology, and Design By Thomas Erl
[2] Software Architecture Patterns by Mark Richards
[3] The Complete List of Indian Cab Ecosystem, features & technologies
[4] Business Intelligence Dashboard: All You Need to Know
[5] Long Polling: A easily implemented concept
[6] Geocoding API overview
[7] Routes API
[8] Leveraging your GPS data using Geospatial analytics