StaySolutionBD (A solution Rental Services Website)

Software Requirement Specification

1. Planning and Research:

- **Market Research**: Understand your target audience, what they want, and how you can make your application stand out from the competition.
- **Determine Technical Requirements:** Understand the tech stack, hosting, and third-party integrations you'll need.

2. Core Features:

- **User Registration and Profiles:**
    - Separate registration processes for property owners and renters.
    - Profiles should store and display relevant details (e.g., owner's properties, tenant's booking history).
- **Property Listing:**
    - Owners can add/edit/delete their property listings.
    - Key details: Images, pricing, amenities, rules, location (with map integration), available dates, etc.
- **Search and Filter**:
    - Tenants can search properties based on various criteria like location, price range, amenities, and more.
    - Detailed property view with all the info and owner contact details.
- **Booking System:**
    - Tenants can book available properties for desired dates.
    - Calendar integration to show availability.
    - Booking confirmation process (either instant or after owner's approval).
- **Payment Integration:**
    - Tenants can pay their rent online.
    - Support various payment methods (credit card, bank transfer, digital wallets).
    - Automatic payment reminders.
- **Refair/Issue Reporting:**
    - Tenants can report issues they face during their stay.
    - Integration with ticketing system so owners can track and resolve the issue.
- **Messaging System**:
    - Direct messaging between owners and tenants.
    - Ensure security and privacy, consider end-to-end encryption.
- **Ratings and Reviews:**
    - After the stay, tenants can rate and review properties.

    - Owners can respond to reviews.

3. Additional Features:

- **Dashboard:** Both for tenants and owners to get a snapshot of bookings, payments, and other relevant details.
- **Notifications**: Email and in-app notifications for bookings, payments, messages, etc.
- **Analytics**: For owners to understand booking trends, popular dates, income tracking, etc.
- **Dynamic Pricing**: Owners can adjust prices based on demand, season, or other factors.

4. Development:

- **Choose the Tech Stack**: Depending on your requirements, you might consider using frameworks and technologies like Next.js for the front end and Node.js, for the backend.
- **Database:** PostgreSQL, could be used to store user profiles, property listings, and booking details.
- **APIs**: Use third-party APIs for payments (e.g., Stripe, PayPal), messaging, maps, etc.

5. Testing:

- Quality Assurance: Ensure that all features work as expected. Test for various user roles (tenant, owner).
- Security: Important due to payments and personal information. Consider regular security audits.

6. Launch and Marketing:

- **Beta Testing**: Launch a beta version for a select group of users to gather feedback and make improvements.
- **Marketing:** Utilize social media, SEO, and possibly partnerships with real estate agents or agencies.

7. Post-Launch:

- Feedback Loop: Keep gathering feedback to refine and enhance the platform.
- Continuous Updates: Regularly update the software based on technological advancements and user requirements.

Entities

User:
- UserID (Primary Key)
- FirstName, LastName, Email, Password, Phone, ProfilePic, UserType, PreferredPropertyType, PreferredAmenities, PreferredLocation, SearchHistory, BookingHistory, SocialMediaLinks, UserStatus

Property:
- PropertyID (Primary Key)
- OwnerID (Foreign Key referencing User)
- Address, Description, NumberOfRooms, Amenities, Rules, Pricing, ImageGallery, AvailabilityCalendar, PropertyStatus, VideoLink, PropertyTags, FloorPlans

Booking:
- BookingID (Primary Key)
- TenantID (Foreign Key referencing User)
- PropertyID (Foreign Key referencing Property)
- StartDate, EndDate, BookingStatus, SpecialRequests

Payment:
- PaymentID (Primary Key)
- BookingID (Foreign Key referencing Booking)
- Amount, PaymentMethod, PaymentStatus, PaymentDate, SecurityDepositAmount

Review:
- ReviewID (Primary Key)
- PropertyID (Foreign Key referencing Property)
- TenantID (Foreign Key referencing User)
- Rating, Comments, ReviewDate, ResponseFromOwner, ReviewPhotos

Issue (Refair):
- IssueID (Primary Key)
- PropertyID (Foreign Key referencing Property)
- TenantID (Foreign Key referencing User)
- IssueDescription, IssueStatus, ReportDate, PriorityLevel

Message:
- MessageID (Primary Key)
- SenderID (Foreign Key referencing User)
- ReceiverID (Foreign Key referencing User)
- Content, Timestamp, MessageCategory

Notification:
- NotificationID (Primary Key)
- UserID (Foreign Key referencing User)
- Content, Timestamp, NotificationType, NotificationPlatform

Wishlist:

- WishlistID (Primary Key)
- UserID (Foreign Key referencing User)
- PropertyID (Foreign Key referencing Property)
- DateAdded

Loyalty Program:

- LoyaltyID (Primary Key)
- UserID (Foreign Key referencing User)
- Points, RedemptionHistory

Local Services & Attractions:

- ServiceID (Primary Key)
- Location, ServiceType, Description, ContactDetails, OperatingHours

Safety:

- SafetyID (Primary Key)
- PropertyID (Foreign Key referencing Property)
- SafetyAmenities, SafetyScore

Marketplace:

- ItemID (Primary Key)
- OwnerID (Foreign Key referencing User)
- ItemDescription, Price, Category

Insurance:

- InsuranceID (Primary Key)
- PropertyID (Foreign Key referencing Property)
- TenantID (Foreign Key referencing User)
- PolicyDetails, CoverageAmount, ExpiryDate

Entity Relationship Diagram Data

1. User:

Attribute	Type
UserID	INT (PK)
FirstName	VARCHAR
LastName	VARCHAR

Email	VARCHAR
Password	VARCHAR
Phone	VARCHAR
ProfilePic	VARCHAR
UserType	ENUM
PreferredPropertyType	VARCHAR
PreferredAmenities	TEXT
PreferredLocation	VARCHAR
SearchHistory	TEXT
BookingHistory	TEXT
SocialMediaLinks	TEXT
UserStatus	ENUM

2. Property:

Attribute	Type
PropertyID	INT (PK)
OwnerID	INT (FK)
Address	TEXT
Description	TEXT

NumberOfRooms	INT
Amenities	TEXT
Rules	TEXT
Pricing	DECIMAL
ImageGallery	TEXT
AvailabilityCalendar	TEXT
PropertyStatus	ENUM
VideoLink	VARCHAR
PropertyTags	TEXT
FloorPlans	VARCHAR

3. Booking:

Attribute	Type
BookingID	INT (PK)
TenantID	INT (FK)
PropertyID	INT (FK)
StartDate	DATE
EndDate	DATE

BookingStatus	ENUM
SpecialRequests	TEXT

4. Payment:

Attribute	Type
PaymentID	INT (PK)
BookingID	INT (FK)
Amount	DECIMAL
PaymentMethod	ENUM
PaymentStatus	ENUM
PaymentDate	DATE
SecurityDepositAmount	DECIMAL

5. Review:

Attribute	Type
ReviewID	INT (PK)
PropertyID	INT (FK)
TenantID	INT (FK)
Rating	INT
Comments	TEXT

ReviewDate	DATE
ResponseFromOwner	TEXT
ReviewPhotos	TEXT

6. Issue (Repair):

Attribute	Type
IssueID	INT (PK)
PropertyID	INT (FK)
TenantID	INT (FK)
IssueDescription	TEXT
IssueStatus	ENUM
ReportDate	DATE
PriorityLevel	ENUM

7. Message:

Attribute	Type
MessageID	INT (PK)
SenderID	INT (FK)
ReceiverID	INT (FK)
Content	TEXT

Timestamp	DATETIME
MessageCategory	ENUM

8. Notification:

Attribute	Type
NotificationID	INT (PK)
UserID	INT (FK)
Content	TEXT
Timestamp	DATETIME
NotificationType	ENUM
NotificationPlatform	ENUM

9. Wishlist:

Attribute	Type
WishlistID	INT (PK)
UserID	INT (FK)
PropertyID	INT (FK)
DateAdded	DATE

10. Loyalty Program:

Attribute	Type
LoyaltyID	INT (PK)
UserID	INT (FK)
Points	INT
RedemptionHistory	TEXT

11. Local Services & Attractions:

Attribute	Type
ServiceID	INT (PK)
Location	TEXT
ServiceType	ENUM
Description	TEXT
ContactDetails	TEXT
OperatingHours	TIME

12. Safety:

Attribute	Type
SafetyID	INT (PK)
PropertyID	INT (FK)

SafetyAmenities	TEXT
SafetyScore	INT

13. Marketplace:

Attribute	Type
ItemID	INT (PK)
OwnerID	INT (FK)
ItemDescription	TEXT
Price	DECIMAL
Category	ENUM

16. Insurance:

Attribute	Type
InsuranceID	INT (PK)
PropertyID	INT (FK)
TenantID	INT (FK)
PolicyDetails	TEXT
CoverageAmount	DECIMAL
ExpiryDate	DATE

Relationships

User to Property:

- Relationship Name: "Owns"
- Description: A user can own multiple properties, but each property is owned by a single user.
- Cardinality: One to Many (from User to Property)

User to Booking:

- Relationship Name: "Books"
- Description: A user (as a tenant) can book multiple properties, but each booking is associated with one user.
- Cardinality: One to Many (from User to Booking)

Property to Booking:

- Relationship Name: "Is Booked As"
- Description: A property can have multiple bookings over time, but each booking is associated with a single property.
- Cardinality: One to Many (from Property to Booking)

Booking to Payment:

- Relationship Name: "Generates"
- Description: Each booking can generate multiple payments, especially if there's a system of installments or split payments.
- Cardinality: One to Many (from Booking to Payment)

Property to Review:

- Relationship Name: "Receives"
- Description: A property can receive multiple reviews, but each review is specific to a single property.
- Cardinality: One to Many (from Property to Review)

User to Review:

- Relationship Name: "Writes"
- Description: A user can write multiple reviews, but each review is written by a single user.
- Cardinality: One to Many (from User to Review)

Property to Issue (Repair):

- Relationship Name: "Has"
- Description: A property can have multiple issues reported, but each issue is associated with a single property.
- Cardinality: One to Many (from Property to Issue)

User to Issue (Repair):

- Relationship Name: "Reports"

- Description: A user can report multiple issues across different properties, but each issue report is made by a single user.
- Cardinality: One to Many (from User to Issue)

User to Message (as a Sender):

- Relationship Name: "Sends"
- Description: A user can send messages to multiple users.
- Cardinality: One to Many (from one User to many Messages)

User to Message (as a Receiver):

- Relationship Name: "Receives"
- Description: A user can receive messages from multiple users.
- Cardinality: One to Many (from one User to many Messages)

User to Notification:

- Relationship Name: "Receives Notification"
- Description: A user can receive multiple notifications.
- Cardinality: One to Many (from User to Notification)

User to Wishlist:

- Relationship Name: "Creates Wishlist"
- Description: A user can have multiple items in their wishlist.
- Cardinality: One to Many (from User to Wishlist)

Property to Wishlist:

- Relationship Name: "Is Wished For"
- Description: A property can be on the wishlist of multiple users.
- Cardinality: One to Many (from Property to Wishlist)

User to Loyalty Program:

- Relationship Name: "Participates In"
- Description: A user can participate in the loyalty program.
- Cardinality: One to One (assuming each user has one loyalty program record)

Property to Safety:

- Relationship Name: "Has Safety Features"
- Description: Each property can have a record of its safety features.
- Cardinality: One to One (each property to its safety record)

User to Marketplace Items:

- Relationship Name: "Lists Item"
- Description: A user can list multiple items in the marketplace.
- Cardinality: One to Many (from User to Marketplace Items)

Property to Insurance:

- Relationship Name: "Is Insured By"
- Description: Each property can have an associated insurance policy.
- Cardinality: One to One (each property to its insurance record)

User to Insurance (as a Tenant):

- Relationship Name: "Purchases Insurance"

- Description: A tenant can purchase multiple insurance policies (especially if they book multiple properties).
- Cardinality: One to Many (from User to Insurance policies)

Model Design

```
enum UserType {

  Owner

  Renter

}


enum PropertyStatus {

  Available

  Booked

  UnderMaintenance

}


enum BookingStatus {

  Confirmed

  Pending

  Cancelled

}


enum PaymentStatus {

  Completed

  Pending

  Failed

}
```

```
enum MessageCategory {

  Inquiry

  BookingRelated

  IssueRelated

}


enum NotificationType {

  Booking

  Cancellation

  Offers

}


enum NotificationPlatform {

  Email

  SMS

  App

}


enum PriorityLevel {

  Low

  Medium

  High

}


model User {

  id                     Int                    @id @default(autoincrement())

  firstName              String

  lastName               String

  email                  String                 @unique

  password               String

  phone                  String

  profilePic             String?
  userType              UserType              @default(Renter)
  preferredPropertyType String?
  preferredAmenities    String[]
  preferredLocation     String?
  searchHistory         String[]
  socialMediaLinks      String[]
  userStatus            String?
  properties            Property[]
  bookings              Booking[]
  reviews               Review[]
  issues                Issue[]
  messagesSent          Message[]             @relation("MessagesSent")
  messagesReceived      Message[]             @relation("MessagesReceived")
  notifications         Notification[]
  wishlists             Wishlist[]
  loyaltyPrograms       LoyaltyProgram[]
  marketplaceItems      Marketplace[]
  insurances            Insurance[]
}

model Property {
  id                     Int           @id @default(autoincrement())
  ownerId                Int
  address                String
  description            String
  numOfRooms             Int
  amenities              String[]
  rules                  String[]
  pricing                Float
  imageGallery           String[]
  availabilityCalendar   DateTime[]
  propertyStatus         PropertyStatus
  videoLink              String?
  propertyTags           String[]
  floorPlans             String?
  owner                  User          @relation(fields: [ownerId], references: [id])
  bookings               Booking[]
  reviews                Review[]
  issues                 Issue[]
  safetyRecord           Safety?
  insurance              Insurance?
}

model Booking {
  id            Int           @id @default(autoincrement())
  tenantId      Int
  propertyId    Int
  startDate     DateTime
  endDate       DateTime
  bookingStatus BookingStatus
  specialRequests String?
  payment       Payment?
  tenant        User          @relation(fields: [tenantId], references: [id])
  property      Property      @relation(fields: [propertyId], references: [id])
}

model Payment {
  id                 Int           @id @default(autoincrement())
  bookingId          Int
  amount             Float
  paymentMethod      String
  paymentStatus      PaymentStatus
  paymentDate        DateTime
  securityDeposit    Float?
  booking            Booking       @relation(fields: [bookingId], references: [id])
}



model Review {
  id                 Int       @id @default(autoincrement())
  propertyId         Int
  tenantId           Int
  rating             Int
  comments           String
  reviewDate         DateTime
  responseFromOwner String?
  reviewPhotos       String[]
  property           Property  @relation(fields: [propertyId], references: [id])
  tenant             User      @relation(fields: [tenantId], references: [id])
}

model Issue {
  id               Int           @id @default(autoincrement())
  propertyId       Int
  tenantId         Int
  description      String
  issueStatus      String
  reportDate       DateTime
  priorityLevel    PriorityLevel
  property         Property      @relation(fields: [propertyId], references: [id])
  tenant           User          @relation(fields: [tenantId], references: [id])
}

model Message {
  id              Int     @id @default(autoincrement())
  senderId        Int
  receiverId     Int
  content        String
  timestamp      DateTime
  category       MessageCategory
  sender         User    @relation("MessagesSent", fields: [senderId], references:
[id])
  receiver       User    @relation("MessagesReceived", fields: [receiverId],
references: [id])
}

model Notification {
  id              Int                 @id @default(autoincrement())
  userId          Int
  content         String
  timestamp       DateTime
  type            NotificationType
  platform        NotificationPlatform
  user            User                @relation(fields: [userId], references: [id])
}

model Wishlist {
  id              Int       @id @default(autoincrement())
  userId          Int
  propertyId      Int
  dateAdded       DateTime
  user            User      @relation(fields: [userId], references: [id])
  property        Property  @relation(fields: [propertyId], references: [id])
}

model LoyaltyProgram {
  id              Int       @id @default(autoincrement())
  userId          Int
  points          Int
  redemptionHistory String[]
  user            User      @relation(fields: [userId], references: [id])
}

model LocalServicesAttractions {
  id               Int     @id @default(autoincrement())
  location         String
  serviceType      String
  description      String
  contactDetails   String
  operatingHours   String
}

model Safety {
  id               Int     @id @default(autoincrement())
  propertyId       Int
  safetyAmenities  String[]
  safetyScore      String
  property         Property @relation(fields: [propertyId], references: [id])
}

model Marketplace {
  id               Int     @id @default(autoincrement())
  ownerId          Int
  itemDescription  String
  price            Float
  category         String
  owner            User    @relation(fields: [ownerId], references: [id])
}

model Insurance {
  id              Int      @id @default(autoincrement())
  propertyId      Int
  tenantId        Int
  policyDetails   String
  coverageAmount  Float
  expiryDate      DateTime
  property        Property @relation(fields: [propertyId], references: [id])
  tenant          User     @relation(fields: [tenantId], references: [id])
}
```

Api end-points

User Endpoints:

Registration:

- POST http://localhost:5000/api/v1/user/users: Register a new user

Login:

- POST http://localhost:5000/api/v1/user/login: User login

User Details:

- GET http://localhost:5000/api/v1/user/users/{id}: Get user details by ID

Update User:

- PUT http://localhost:5000/api/v1/user/users/{id}: Update user details

Delete User:

- DELETE http://localhost:5000/api/v1/user/users/{id}: Delete a user

Property Endpoints:

Add Property:

- POST http://localhost:5000/api/v1/property/properties: Add a new property

Get All Properties:

- GET http://localhost:5000/api/v1/property/properties: Get all properties

Property Details:

- GET http://localhost:5000/api/v1/property/properties/{id}: Get property details by ID

Update Property:

- PUT http://localhost:5000/api/v1/property/properties/{id}: Update property details

Delete Property:

- DELETE http://localhost:5000/api/v1/property/properties/{id}: Delete a property

Booking Endpoints:

Create a Booking:

- POST http://localhost:5000/api/v1/booking/bookings: Create a new booking

Get Booking Details:

- GET http://localhost:5000/api/v1/booking/bookings/{id}: Get booking details by ID

Get Bookings for a User:

- GET http://localhost:5000/api/v1/user/users/{id}/bookings: Get bookings for a specific user

Update Booking:

- PUT http://localhost:5000/api/v1/booking/bookings/{id}: Update a booking

Cancel Booking:

- DELETE http://localhost:5000/api/v1/booking/bookings/{id}: Cancel a booking

Payment Endpoints:

Make Payment:

- POST http://localhost:5000/api/v1/payment/payments: Make a new payment

Get Payment Details:

- GET http://localhost:5000/api/v1/payment/payments/{id}: Get payment details by ID

Payments for a Booking:

- GET http://localhost:5000/api/v1/booking/bookings/{id}/payments: Get payments for a specific booking

Review Endpoints:

Add a Review:

- POST http://localhost:5000/api/v1/review/reviews: Add a review for a property

Get Reviews for a Property:

- GET http://localhost:5000/api/v1/property/properties/{id}/reviews: Get all reviews for a specific property

Issue (Repair) Endpoints:

Report an Issue:

- POST http://localhost:5000/api/v1/issue/issues: Report an issue for a property

Get Issue Details:

- GET http://localhost:5000/api/v1/issue/issues/{id}: Get issue details by ID

Update an Issue:

- PUT http://localhost:5000/api/v1/issue/issues/{id}: Update an issue (e.g., status change)

Message Endpoints:

Send a Message:

- POST http://localhost:5000/api/v1/message/messages: Send a message

Get Messages for a User:

- GET http://localhost:5000/api/v1/user/users/{id}/messages: Get all messages for a specific user

Notification Endpoints:

Create a Notification:

- POST http://localhost:5000/api/v1/notification/notifications: Create a new notification for a user

Get Notifications for a User:

- GET http://localhost:5000/api/v1/user/users/{id}/notifications: Fetch all notifications for a specific user

Delete a Notification:

- DELETE http://localhost:5000/api/v1/notification/notifications/{id}: Remove a specific notification

Wishlist Endpoints:

Add Property to Wishlist:

- POST http://localhost:5000/api/v1/wishlist/items: Add a property to a user's wishlist

View User Wishlist:

- GET http://localhost:5000/api/v1/user/users/{id}/wishlist: Get all properties in a user's wishlist

Remove Property from Wishlist:

- DELETE http://localhost:5000/api/v1/wishlist/items/{id}: Remove a specific property from the wishlist

Loyalty Program Endpoints:

Join Loyalty Program:

- POST http://localhost:5000/api/v1/loyalty/enroll: Enroll a user in the loyalty program

Get User Loyalty Points:

- GET http://localhost:5000/api/v1/user/users/{id}/loyalty: Fetch loyalty points and redemption history for a user

Redeem Loyalty Points:

- POST http://localhost:5000/api/v1/loyalty/redeem: Redeem points for rewards or discounts

Local Services & Attractions Endpoints:

List a Local Service/Attraction:

- POST http://localhost:5000/api/v1/services/list: List a new local service or attraction

Get Services & Attractions by Location:

- GET http://localhost:5000/api/v1/services?location={location}: Fetch services and attractions based on a location

Safety Endpoints:

Add Safety Record for Property:

- POST http://localhost:5000/api/v1/safety/records: Add a safety record for a specific property

Fetch Safety Records for Property:

- GET http://localhost:5000/api/v1/property/properties/{id}/safety: Get safety records for a property

Marketplace Endpoints:

List an Item in Marketplace:

- POST http://localhost:5000/api/v1/marketplace/list: List an item or service in the marketplace

Browse Marketplace by Category:

- GET http://localhost:5000/api/v1/marketplace?category={category}: View items in the marketplace by category

Insurance Endpoints:

Purchase Insurance:

- POST http://localhost:5000/api/v1/insurance/purchase: Buy insurance for a property booking

View Insurance Policies for User:

- GET http://localhost:5000/api/v1/user/users/{id}/insurance: View all insurance policies bought by a user

Login Endpoint:

User Login:

- POST http://localhost:5000/api/v1/user/login: User login

JSON DATA:

User Endpoints

1. Register User

POST /api/v1/user/users

```
	{
```

```
          "firstName": "John",
          "lastName": "Doe",
          "email": "john.doe@example.com",
          "password": "securePassword123",
          "phone": "123-456-7890",
          "profilePic": "link_to_profile_pic.jpg",
          "userType": "Renter",
          "preferredPropertyType": "Apartment",
          "preferredAmenities": ["WiFi", "Swimming Pool"],
          "preferredLocation": "Downtown",
          "searchHistory": ["Property1", "Property2"],
          "socialMediaLinks": ["facebook_link", "twitter_link"],
          "userStatus": "Verified"
        }
```

2. User Login

POST /api/v1/user/login

```
        {
          "email": "john.doe@example.com",
          "password": "securePassword123"
        }
```

Property Endpoints

POST /api/v1/property/properties

```
        {
          "ownerID": 1,
          "address": "123 Main St, City, Country",
          "description": "A beautiful 2-bedroom apartment in the city center.",
          "numberOfRooms": 3,
          "amenities": ["WiFi", "Swimming Pool", "Parking"],
          "rules": "No smoking. No pets.",
          "pricing": 1200,
          "availabilityCalendar": {
            "Jan": ["1", "2", "5-20"],
            "Feb": ["1-10"]
          },
          "propertyStatus": "Available",
          "videoLink": "link_to_video.mp4",
          "propertyTags": ["sea-view", "family-friendly"],
          "floorPlans": "link_to_floorplan.jpg"
        }
```

Booking Endpoints

POST /api/v1/booking/bookings

```
{
  "tenantID": 1,
  "propertyID": 2,
  "startDate": "2023-11-01",
  "endDate": "2023-11-10",
  "bookingStatus": "Confirmed",
  "specialRequests": "Need an extra bed"
}
```

Payment Endpoints

POST /api/v1/payment/payments

```
{
  "bookingID": 3,
  "amount": 1200,
  "paymentMethod": "Credit Card",
  "paymentStatus": "Paid",
  "paymentDate": "2023-10-15",
  "securityDepositAmount": 200
}
```

Review Endpoints

POST /api/v1/review/reviews

```
{
  "propertyID": 2,
  "tenantID": 1,
  "rating": 4,
  "comments": "Great place. Loved the amenities.",
  "responseFromOwner": "Thank you for your feedback!"
```

```
        }
```

Issue (Refair) Endpoints

POST /api/v1/issue/issues

```
        {
          "propertyID": 2,
          "tenantID": 1,
          "issueDescription": "The kitchen sink is leaking.",
          "issueStatus": "Reported",
          "reportDate": "2023-11-05",
          "priorityLevel": "Medium"
        }
```

Message Endpoints

POST /api/v1/message/messages

```
        {
          "senderID": 1,
          "receiverID": 2,
          "content": "Hi, I'd like to inquire about the availability in December.",
          "timestamp": "2023-10-15 14:32:15",
          "messageCategory": "Inquiry"
        }
```

Notification Endpoints

POST /api/v1/notification/notifications

```
        {
          "userID": 1,
          "content": "Your booking for Property 2 has been confirmed!",
          "timestamp": "2023-10-15 15:10:05",
          "notificationType": "Booking Confirmation",
          "notificationPlatform": "Email"
        }
```

Wishlist Endpoints

POST /api/v1/wishlist/wishlists

```
{
  "userID": 1,
  "propertyID": 3,
  "dateAdded": "2023-10-12"
}
```

Loyalty Program Endpoints

POST /api/v1/loyalty/loyalties

```
{
  "userID": 1,
  "points": 150,
  "redemptionHistory": [{
    "date": "2023-10-01",
    "pointsRedeemed": 50,
    "reward": "10% Discount"
  }]
}
```

Local Services & Attractions Endpoints

POST /api/v1/localServices/services

```
{
  "location": "Near Property 2",
  "serviceType": "Restaurant",
  "description": "A quaint little Italian place with a wood-fired pizza oven.",
  "contactDetails": "123-456-7890",
  "operatingHours": "10am - 10pm"
}
```

Safety Endpoints

POST /api/v1/safety/safeties

```
{
  "propertyID": 2,
  "safetyAmenities": ["Fire extinguisher", "Smoke alarm", "First Aid Kit"],
  "safetyScore": 8.5
}
```

Marketplace Endpoints

POST /api/v1/marketplace/items

```
{
  "ownerID": 1,
  "itemDescription": "Mountain bike for rent",
  "price": 15,
  "category": "Outdoor Equipment"
}
```

Insurance Endpoints

POST /api/v1/insurance/policies

```
{
  "propertyID": 2,
  "tenantID": 1,
  "policyDetails": "Comprehensive coverage including damage, theft, and natural
disasters.",
  "coverageAmount": 50000,
  "expiryDate": "2024-10-15"
}
```