**UWM – Architectural Style**

**Architecture: Overview**

Micro services architecture is a design pattern where the application is built as a collection of small, independent services that are focused on specific business functionalities. Each Micro Service is responsible for a single task and can be developed, deployed, and scaled independently. This approach offers several benefits, such as:

1. Scalability: Each Micro Service can be scaled independently based on its specific needs, allowing for better resource utilization and improved performance.
2. Modularity and Maintainability: Micro Services are loosely coupled, making it easier to modify and maintain individual services without affecting the entire application.
3. Independent Deployment: Each Micro Service can be deployed independently, enabling faster release cycles and reducing the risk of impacting the entire application during updates.
4. Technology Diversity: Micro Services allow you to use different technologies and frameworks for each service, depending on the specific requirements and preferences.
5. Team Autonomy: Micro Services promote smaller, cross-functional teams that can work independently on different services, enabling faster development cycles and greater agility.
6. Fault Isolation: Issues in one Micro Service generally don't affect other services, providing better fault isolation and reducing the impact of failures.

**List of potential Micro Services:**

1. Borrower Service: Responsible for managing borrower information, including creating, retrieving, updating, and deleting borrower records.
2. Loan Application Service: Handles loan application-related operations, such as creating, retrieving, updating, and deleting loan applications.
3. Employment Service: Manages employment details for borrowers, including adding, retrieving, updating, and deleting employment records.
4. Asset Service: Handles asset-related operations for borrowers, including adding, retrieving, updating, and deleting asset records.
5. Liability Service: Manages borrower liabilities, including adding, retrieving, updating, and deleting liability records.
6. Credit Report Service: Deals with credit reports, including retrieving and updating credit scores and history for borrowers.
7. Rate Lock Service: Manages rate lock information for loan applications, including creating, retrieving, updating, and deleting rate lock records.

To connect and call the Micro Services, you can use a combination of API endpoints and communication protocols such as HTTP/HTTPS and RESTful APIs.

Here's a sample scenario:

1. The client application makes an HTTP request to the Borrower Service API to create a new borrower record.
2. The Borrower Service API receives the request, validates the data, and stores the borrower information in its database.
3. Upon successful creation of the borrower record, the Borrower Service API responds with a success status code and the ID of the newly created borrower.
4. The client application then needs to retrieve the borrower's employment information. It makes an HTTP request to the Employment Service API, providing the borrower ID as a parameter.
5. The Employment Service API receives the request, retrieves the employment details associated with the borrower ID from its database, and responds with the requested information.
6. The client application continues this process to interact with other Micro Services, such as Asset Service, Liability Service, Credit Report Service, and Rate Lock Service, by making HTTP requests to the corresponding APIs and receiving responses with the requested data.

The communication between the Micro Services can be achieved through HTTP requests using RESTful APIs, which provide a standard and scalable way to exchange data. Each Micro Service exposes a set of API endpoints that can be called by other Micro Services or client applications. The data is typically exchanged in JSON format.

It's important to design the APIs with proper authentication, authorization, and error handling mechanisms to ensure data security, access control, and reliable communication between the Micro Services. Additionally, service discovery and load balancing techniques can be employed using Ribbon , Eurekha Server to facilitate service communication and distribution of incoming requests among multiple instances of Micro Services for better scalability and availability.