**Project Overview:**

The project involves building a decentralized payment processor using a Flask API that allows users to send and receive payments using a cryptocurrency. The backend handles user accounts and transactions, and it's divided into two parts: Part 1 focuses on using MongoDB for account management, and Part 2 integrates with the XRPL (XRP Ledger) for cryptocurrency transactions.

**Project Structure:**

**Folder Structure:**

**app**: The main application folder.

**\_\_init\_\_.py**: Initializes the Flask app, MongoDB, and registers blueprints.

**config.py**: Contains configuration settings for the app.

**routes.py**: Defines the API endpoints and their functionality.

**models.py**: Defines the data model for users, including MongoDB setup.

**Project Flow:**

**run.py:** Entry point of the application, where the Flask app is run.

**Part 1:** Flask API + MongoDB:

User registration with username, email, and password.

User login with JWT authentication.

Retrieval of account balance for logged-in users.

Sending cryptocurrency to other users, updating balances in MongoDB.

**Part 2: Flask API + XRPL:**

Extended functionality to send XRP using the XRPL.

Integration with XRPL Python library.

Handling XRPL testnet transactions.

Updating user account balances in MongoDB after XRPL transactions.

**Project Workflow:**

- The user can register with their username, email, and password. User data is stored in MongoDB with password encryption.

- Registered users can log in using their username and password. JWT is used for authentication.

- Users can check their account balance by making a GET request to the /balance endpoint.

- Users can send cryptocurrency to another user by specifying the recipient's username and the amount using the /send payment endpoint.

- In Part 2, the project extends to send XRP through the XRPL. This involves creating a new endpoint, /xrpl\_send\_payment, which interacts with the XRPL Python library.

- The XRPL testnet is set up, and proper error handling is implemented for failed transactions.

- User account information in MongoDB is updated to reflect changes in account balance after XRPL transactions.

**Technologies Used:**

**Flask:** For building the API.

**MongoDB:** For storing user account information.

**XRPL Python library:** For XRPL interactions.

**JWT:** For user authentication.

**Conclusion:**

This project represents a decentralized payment processor with two main parts. It provides user registration, login, balance retrieval, and the ability to send cryptocurrency, integrating both MongoDB and XRPL. This structure facilitates secure, decentralized payment processing for users.