University of Cape Town ECO5040S - Financial Software Engineering - Class Project Brief - XRP Ledger Crypto Asset Telegram Bot

Release Date: 2025-08-29
Due Date: 2025-09-26

Background

Crypto assets are financial resources that are minted and exchanged via blockchain technology, and directly controlled through public-private key cryptography.[1] They can take the form of native cryptocurrencies (e.g. BTC, XRP, XLM) or stablecoins (e.g. RLUSD, USDC, USDT). While crypto assets have historically been met with institutional and regulatory scepticism and scrutiny, as of 2025, they are entering a pivotal phase, in particular with institutional adoption accelerating, and regulatory clarity emerging.

Retail adoption of crypto assets can be accelerated by provision of crypto asset services to consumers by existing financial services providers. Financial services providers such as Robinhood[2], have stepped up to this market need, and are providing crypto asset services, and also building proprietary blockchain infrastructure[3] to cement their position in the battle for the future of payments. Another approach to accelerating retail adoption of crypto assets is by layering crypto asset payments on top of existing and ubiquitous instant messaging applications such as WhatsApp Messenger (WhatsApp) and Telegram Messenger (Telegram). Telegram in particular, is a cloud-based, cross-platform instant messaging service that allows users to exchange messages and media.[4] It is simple to use, fast, secure, and one of the most downloaded applications in the world, with over 1 billion active users. Telegram offers a Bot API which allows developers to connect custom bots - interfaces for source code running on a server, with communication managed via a simple HTTPS-interface - to Telegram.[5]

This brief outlines the purpose of the class project, which is to layer XRP Ledger-based crypto asset payments on top of the Telegram messaging infrastructure.

---

[1] Digital assets in payments and transaction banking. https://www.resbank.co.za/en/home/publications/publication-detail-pages/working-papers/2024/digital-assets-in-payments-and-transaction-banking
[2] https://robinhood.com/us/en/
[3] https://robinhood.com/us/en/blockchain/
[4] https://telegram.org/
[5] https://core.telegram.org/api#bot-api

Objective

The objective of this project is to design and build a Telegram Bot that is integrated with the XRP Ledger[6] to allow users to send and receive XRP, and access real-time market prices. The Bot will be accessed via the Telegram mobile application, and be connected to a Python-based backend(s). Your Bot application is expected to be implemented using a modular architecture style, utilising a Python Framework like Flask, Django or FastAPI; and a relational SQL database (e.g. SQLite, Postgres, MySQL).

Specifically, the Bot should:

1. Be accessible via a Telegram Bot interface.
2. Integrate with Telegram using the Telegram Bot API (and specifically making use of webhooks instead of long polling, to get updates from Telegram).
3. Integrate with the XRP Ledger TestNet via the xrp-py[7] Python library.
4. Allow users to sign up using their Telegram accounts.
5. The Telegram Bot will implement a custodial model i.e. user XRPL accounts will be generated and stored server-side. On sign up:
    - An XRPL account should be generated for a user, and the secret key must be encrypted and stored in your database.
    - A user must be allocated some XRP.
6. Display the user's balances on XRP Ledger (balances of XRP and optionally any other XRPL-assets).
7. Allow a user to send XRP to another user via the Bot.
8. Allow a user to view historical XRP prices obtained from an historical crypto asset price API.

Deliverables

The project deliverables include:

1. Technical specification document. This document should include the following:
    - Functional requirements
    - UI mockups
    - Implementation
    - Technology justification
    - Architecture diagram(s)
    - Sequence diagram(s)
    - How it works

[6] https://xrpl.org/
[7] https://xrpl.org/docs/tutorials/python/build-apps/get-started

    - Evaluation including reconciliation of functional requirements, and performance tests to simulate user sign ups and peer-to-peer payments.
2. Proof-of-Concept (POC) application (including GitHub repo and deployed version).
3. Technical presentation. This will accompany the technical specification and POC. Each presentation will be 20 minutes in duration (15 minutes for the presentation & the live demo, and 5 minutes for questions). You should aim to present everything in a maximum of 10 - 12 slides. Your presentation should include the following:
    - Title, authors and affiliations
    - Background / Motivation / Problem area
    - Objectives of the project
    - Design and implementation of the solution
    - Outcomes / How it works and Evaluation
    - Conclusion & Future work
    - Appendix with further technical details, anticipated responses to questions etc. (optional)

The project will culminate in a presentation and a live demo by each team.

Additional Considerations

- Useful pointers for Telegram Bots:
    - Commands are identified by the leading forward slash (e.g., /start, /help, /settings).
    - Telegram recommends supporting certain global commands for a consistent user experience (e.g., /start, /help).
    - Examples of common commands:
        - /start: Often used to initiate interaction with a bot and display a welcome message or main menu.
        - /help: Provides information about the bot's functionality and lists available commands.
        - /settings: Allows users to customize Bot settings or preferences.
        - /cancel: Used to cancel an ongoing operation or interaction.
- To get free XRP on the XRP Ledger TestNet, make use of faucets. (you can get some free XRP from a TestNet faucet https://xrpl.org/resources/dev-tools/xrp-faucets).
- Historical prices of crypto assets can typically be obtained for free from crypto exchanges.

- You can deploy the Python backend(s) for your Bot to cloud providers such as Render[8], Railway[9] etc.
- Consider generating synthetic users for your performance tests and simulations.

Useful Resources

Additional technical resources:

- https://help.xaman.app/app/learning-more-about-xaman/how-to-access-testnet-on-xrp-ledger
- https://www.freecodecamp.org/news/how-to-create-a-telegram-bot-using-python/
- https://xrpl.org/resources/dev-tools/xrp-faucets
- https://xrpl.org/docs/tutorials/python/build-apps/get-started

Support

- Hub space - Project teams will have access to the Hub as a co-working space, Tuesday - Thursday, 9AM - 6PM.
- Technical support hours - The technical specialists from the Hub will be available Tuesday - Thursday, 11AM - 1PM, for the duration of the project, in-person / MS Teams.
- Project check-ins - There will be bi-weekly check-in's on Fridays on MS Teams.

Teams

The class will be split into 3 teams of up to 4 students each. These will be communicated via email when the project is released.

Dates

Project dates below:

- Friday 29th August - Project brief released & team allocations.
- Monday 1st September - Project kick-off meeting (MS Teams).
- Friday 5th September - Project check-in (MS Teams).
- Saturday 6th September - Mid-term vacation starts.
- Sunday 14th September - Mid-term vacation ends.
- Friday 19th September - Project check-in (MS Teams).
- Friday 26th September - Project complete and final presentation hand-in.
- Monday 29th September - Project presentations and demos.

[8] https://render.com/
[9] https://railway.com/

Evaluation

The ECO5040S - class project constitutes 25% of the final mark for the ECO5040S Financial Software Engineering course (i.e. 25% research paper presentation, 25% class project & 50% final exam). Each student will be graded using the grading rubric on the next page. The peer-evaluation forms will be sent out via email.

Criteria	**Notes**	**Points**
Peer-evaluation (average of all peer grades)		/5
Final presentation (group grade assigned by lecturer)		/5
Final Proof-of-Concept (group grade assigned by lecturer)		/10
General comments		
	Total	/20

Questions can be sent to Julian Kanjere (University of Cape Town) - julian.kanjere@uct.ac.za