İSTANBUL BEYKENT ÜNİVERSİTESİ

Bil 401 _ Siber Güvenlik ve Büyük Veri
Fall 2025/2026
HW

The Smart City Automation with Digital Banking Integration system is a multifaceted platform aimed at enhancing urban living through technology-enabled automation, monitoring, and secure online transactions. This system incorporates an array of functionalities such as remote control of city infrastructure, environmental monitoring, security notifications, and a digital banking component that accepts both fiat and cryptocurrency payments. The goal is to streamline city management, improve resident convenience, and ensure public safety through intelligent system automation and user-friendly interfaces.

Problem Statement:

In this HW, you are designing a comprehensive Smart City automation system with focus on Digital Banking system capable of handling both traditional currency and cryptocurrency payments. In this case, you are considering security features of your smart-city system by design during initial phases.

- **Horizontal view of targeted system**: This Smart City system oversees public utilities, street lighting, traffic management, and public safety by leveraging sensors, cameras, and environmental monitors. Residents can remotely control home devices, view city services, and receive alerts for any unusual activities in their neighborhood. The Digital Banking system facilitates secure transactions for city services (e.g., parking fees, public transit tickets) and supports cryptocurrency payments, enabling a modern, flexible payment infrastructure. Additionally the city has secure login requirement for the digital banking transaction for all amounts including quantum-resistant encryption and dynamic countermeasure features also.

- **Smart City System Main Features**: In the design of a use case for Smart City Automation integrated with digital banking, key primary actors include Residents, the CityController, and BankingService. Residents can remotely manage their home devices, receive security notifications, and conveniently view and pay for city services, such as parking and utilities through a digital banking platform that supports cryptocurrency transactions. The CityController oversees all city infrastructure, managing street lights, traffic signals, and public safety sensors, while also coordinating routines like deactivating street lights at sunrise and processing resident transactions.

    - The BankingService is crucial for facilitating both fiat and cryptocurrency payments, ensuring secure online transactions, and sending transaction notifications. Secondary actors include Public Safety Authorities, who receive real-time alerts of emergencies or security breaches from the city's sensors, and Public Utility Services, which monitor and manage utilities like water and electricity, responding swiftly to maintenance needs indicated by sensor alerts. This interconnected ecosystem enhances city management, promotes resident engagement, and streamlines urban services through advanced technology.
    - Some of the sample digital trading system clients (for TRNC, - https://www.girince.com/ - https://nesdersan.com/ - https://technopluskibris.com/ - https://www.digikeycomputer.com/ - https://www.kibrissanalmarket.com/
    - some global samples , - https://www.akakce.com/ - https://www.hepsiburada.com/ - https://www.amazon.com.tr/ - https://www.alibaba.com/).

The system introduces:

- **Hybrid-Cloud Integration**: Centralized data storage and distributed analytics for real-time processing.
    - **Hybrid-Cloud Features**
        - Centralized database for IoT sensor data.
        - Scalable analytics engine for processing real-time and historical data.
        - Secure API endpoints for mobile and web applications.
- **Advanced Security Measures**: Multi-factor authentication, role-based access control, and encrypted communications.
- **Dynamic GUI**: Interactive dashboards for administrators and customizable views for residents.
    - **Predictive Trusted Analytics**: **Trusted AI-driven** insights to forecast traffic congestion, energy needs, and security alerts.

Task 1: Students are tasked with developing a modular, scalable system using design patterns to ensure ease of control, adaptability, and real-time responsiveness. This case study challenges the system to solve complex issues, such as managing interconnected services, ensuring secure and timely transactions, and coordinating automated routines for multiple entities within a city framework.

A. Analyze potential security threads briefly and identify thread profile of your DigiBank digital banking system in 1 or 2 paragraphs. Draw a detailed UML Class/system diagram, write pseudo code and draw mock-up GUI of your defense system to illustrate your design.
B. Deploy DigiBank Alpha hybrid-cloud package in a public cloud platform and observe the risks.
C. Deploy kali linux (**https://www.kali.org/**) and implement some of the basic attacks including DDoS attack and attack vectors to the deployed sample demo DigiBank hybrid-cloud system.

Task 2:

A. Write 1 paragraph research abstract for potential security threads and potential research directions briefly. Identify thread profile **defense methodology** of your DigiBank digital banking system and implement within your draft code/GUI.
B. Extend your abstract and technical report in form of conference or a journal paper for a few pages by merging your related inputs in IEEE conference template format. You can target Class A conference or Q1 journal submission also.

Make screen shots of Task 1 and 2 all items, include in your report.

Suggested Key Patterns:

- **Singleton Pattern** for the CityController to centralize management.
- **Command Pattern** for commands such as turning on/off street lights or processing payments.
- **Observer Pattern** to send notifications for security events or city-wide updates.
- **Adapter Pattern** to integrate multiple crypto currencies into the digital banking system.
- **Template Method Pattern** for automating daily routines in city services, like lighting schedules or security checks.

Deliverables

1. **UML Class/System Diagram object interactions, pseudo code, and mock-up GUI**: Detailing interactions between actors, system, classes and objects (Task.1.A).
2. **GUI with selected performance monitoring metrics**: Designed using Figma, papyrus, UML Lab or a similar tool for automated code-generation from diagram (Task 1.B and Task 1.C).
3. **Code Implementation**: A functional prototype with key features implemented (Task 2.A)

4. **Test Results and research paper (conference/journal)**: Evidence of system functionality, including screenshots of GUI and output in a conference or journal template format (Task. 2.B).
5. **Report**: Covering all details including the architecture, patterns, cloud integration, and challenges (Task.2.B).

Submission Files

1- **Full report:** in pdf format includes all details (studentID_fullName.pdf). Try to minimize size and content, write it in a research paper format of IEE conference template.
2- **Zip File:** Please submit only one Zip File (size<=5mb) (studentID_fullName.zip) that includes the following components other than the full report: **Source code**, **UML Diagram**, and **Screenshots of Implemented Code and executed application**. Wrong naming causes -1 point in HW grade.

Grading

This HW 1 is worth 25% of your overall course grade as follows:

Grading Item (Deliverables)	**Weight**
D.1 Smart-government defense system UML Class Diagram Object interactions with holistic view, pseudo code, and mock-up GUI**.**	20%
D.2 GUI with performance metrics.	20%
D.3 Code Implementation (JAVA/Python)	30%
D.4 Research paper submission (conference of journal)	10%
D.5 Full Report	20%
Bonus Points for Running DigiBank with registered users.	10 points

<u>Bonus Points Requirements:</u>

Good news, your critical bank client has decided to use DigiBank application since it utilizes secure by nostalgy methodology perfectly. It makes it secure and more than sufficient for the bank requirements. So that to get the bonus;

1- **You have to run Digibank application after that export txt file with today's time stamp.**
2- **Save a few users with the DigiBank app and utilize observer pattern to email yourself the exported txt file by DigiBank app. You have the monitor the ./txt path of the DigiBank app to be able to e-mail yourself in real-time. Make screenshot of the DigiBank running app, .txt file, e-mail file with time labels to see it is emailed to yourself in real-time.**

Important Notes

- **Use Case Diagram:** Please create a detailed use case diagram that visually represents the functionalities and interactions within the Smart City Automation system integrated with Digital Banking. This diagram should illustrate the various actors, their roles, and how they interact with the system.
- **Clarifying Code Complexity**: The project description outlines the comprehensive features and complexities of the Smart City system, the code should serves as a simplified representation of these structures. It is essential to understand that the implementation is meant to demonstrate how these concepts can be organized and executed within a software framework.
- **Focus on Design Patterns**: Pay particular attention to how the design patterns are applied in the code. Ensure you understand their purpose and how they facilitate system scalability, maintainability, and responsiveness.
- **For Code Implementation it is highly recommended to use JAVA language**.
- **If you are using other patterns, then suggested adding logical rationale for that**.
- **Make sure to add comments in your code**.
- **For File submissions**:
- Please submit only one zip file (studentID_fullName.zip) (size<=5mb) that includes the following components: **Pseudocode**, **UML Use Case Diagram**, **Implemented Code**, **Screenshots of Implemented Code, Sources of Complexity** and **Proposed Modular Solution**.
- Make sure to upload the report to the Beykent pusula submission link, ensuring that all images are clear and readable.
- **Suggested Tools**
  For Use case:
  Draw.io (https://app.diagrams.net/)
  Figma (https://www.figma.com/)
  Online Compilers for JAVA/Python:
  - Programiz (https://www.programiz.com/java-programming/online-compiler/)
  - Online Java Compiler IDE (https://www.jdoodle.com/online-java-compiler)
  - Programiz (https://www.programiz.com/python-programming/online-compiler/)
  - Online Python Beta (https://www.online-python.com/)

  Offline Tools:
  - How to Install Python on Mac (https://docs.python.org/3/using/mac.html)
  - How to install Python on Windows (https://docs.python.org/3/using/windows.html)
  - Eclipse (JAVA): (https://www.eclipse.org/)
  - Modelio (https://github.com/ModelioOpenSource/Modelio/releases/tag/v5.4.1) , https://sourceforge.net/projects/modeliouml/
  - **KALI LINUX https://www.kali.org/ , for pen-testing and general security trials.**