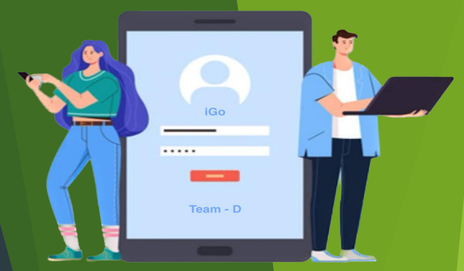
iGo: Ticket Vending Machine

SOEN 6461 - SOFTWARE DESIGN METHODOLOGIES - WINTER 2023

Team - D

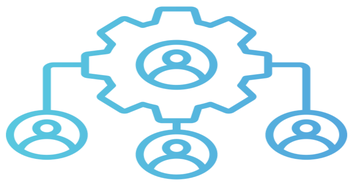
Apekshaba Gohil | Pratik Gondaliya | Carlos Garcia | Amro Elbahrawy| Ernesto Garsiya

Yashwanth Gundlapally

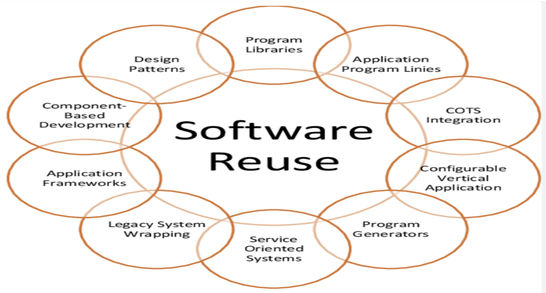


Collaboration patterns

- The main collaboration patterns that were used in the project were **group meetings**
- To make everyone notified of any changes we used in the project were **group meetings** collaborated on **Google Docs** and also maintained our work updated on **GitHub.**
- Meetings were conducted at the beginning of each deliverable **to discuss the deliverable problems, solutions, and confusions**.
- During the work on iGo project we followed Parallel Collaboration.



Reuse Potential



- **Total Reuse:** An iGo smart card system can be relocated to a new location to provide contactless payment services.
- **Partial Reuse:** RFID chips or readers maybe removed and used in other applications.
- **Upcycling:** The smart cards used can also be upcycled to create new products with different functions or designs.
- **Recycling:** Cannot be reused in any form, can still be recycled to recover valuable materials.

Lessons Learnt

- **Communication and collaboration are key**: Effective communication and collaboration are essential for the success of any software development project.
- **Prioritize the most important features:** With limited time and resources, it's critical to focus on the most important features of a software system. The team identified the top use cases and prioritized them based on their impact on the user experience.
- **Use the right tools:**Choosing appropriate tools for the job and being open to new ones is essential, as demonstrated by the team's effective use of PlantUML and LaTeX for diagramming and report writing.
- **Iterative is Key:**Iterative refinement of the design based on feedback and testing is key to improving the overall quality of the software development process.

Scope in different Programming Language

- Solution domain model uses standard OOD principles.
- Syntax-agnostic.
- No language-specific features or constructs (e.g., no multiple inheritance). Strong typed attributes (data types supported by most OOD languages).
- Methods have standard signatures (in modern OOD languages).

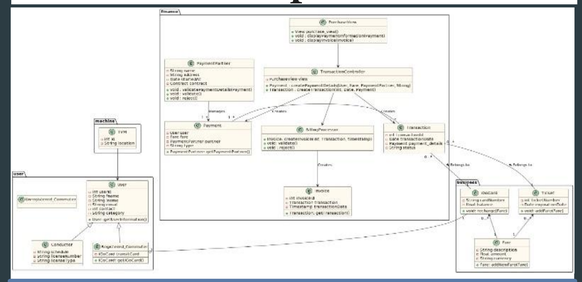
Critical Decisions

- **Designing the class diagram and domain model:** To determine the system's structure and component's interaction.
- **Coordinating communication and collaboration:** Effective communication and collaboration was a key to smooth work over the project tenure.
- **Choosing the right diagramming tool:** As it effectively visualized and communicated the system design, and ability to generate high-quality output.
- **Choosing the right technology stack:** To avoid poor performance, security vulnerabilities, and maintenance and scalability issues.

Limitations

- **Limited Design Options of Tkinter**
- **Limited Platform Support:** includes only
- desktop applications.
- **Limited Graphics Capabilities**
- **Performance:** May not be a good fit for real-time applications.
- **Lack of Compatibility:** Tkinter may not be compatible with other Python GUI libraries.

Insulated Implementation



- **Modular Approach.**
- **Well-defined interfaces interacting without loss of privacy.**
- **MVC Design pattern to isolate GUI, Controller and Database for easy updation.**
- **Module testing.**
- **Use of versioning control tool such as Git**

References

- PANKAJ KAMTHAN (2023) "Introduction To Domain Modeling" - Section 14,15,16.
  PANKAJ KAMTHAN (2023) "Introduction To Use Case Modeling" - Section 11, 12.
- http://www.stm.info/en

Acknowledgements

We would like to express our deepest gratitude to Prof Pankaj Kamthan and TA's of this course, Thank you for your unwavering commitment to your students, and for inspiring us to strive for excellence in all our academic pursuits.

Tools Used

UML Designer · IJ · Tkinter · jupyter