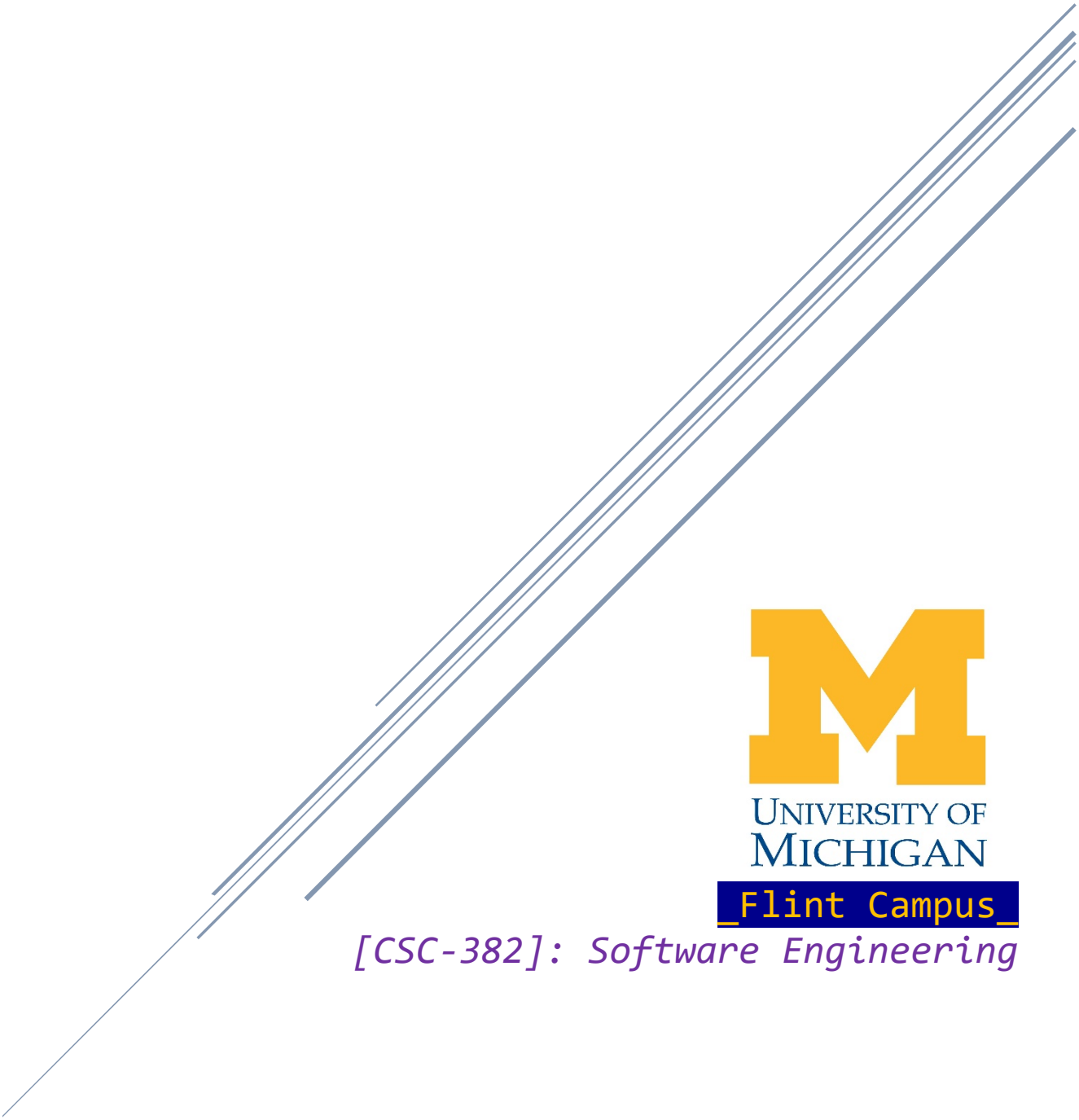
FINAL EXAM

Cason Konzer: 8/12-16/2021



UNIVERSITY OF MICHIGAN
Flint Campus

[CSC-382]: Software Engineering

Contents

6.0 **Class Relationships**: Pg. [28-29]

6.1 Develop a singular, or individual, class diagram(s)and include all the following relationships amongst your classes.

7.0 **Sequence Diagram/Charts and State Diagrams**: Pg. [31-34]

7.1 Develop two (2) Sequence Diagrams/Charts.
7.2 Develop two (2) State Diagrams.

8.0 **Components**: Pg. [36]

8.1 Define one (1) component and draw the classes that would be part of this component and be sure to identify the methods that would serve as the interfaces to this component.

9.0 **Testing**: Pg. [38-40]

9.1 Describe in detail how the application will be tested –this will include which testing scenarios will be utilized (regression and/or standard scenarios) and what will constitute successful tests and overall testing success. A complete testing plan should be at least three pages in length with no obvious gaps in the test plan and the included test scenarios.

[Note]:

-A page is defined as single space, 12-point font, with one inch or less side margins, and 1.5 inch or less top and bottom margins. Write in complete sentences and provide references with citations as needed.

-The Final Exam is worth 130 points. All work must be submitted electronically as one word/pdf file. Due 8/16.

This page is intentionally left Blank.

Project Case

Smart Vending Machine Case Study

You have been tasked with developing the software for the **next generation vending machine** for use at universities.

The vending machine has several unique features that must be developed for the first time in such an application. These features include:

1. The vending machine must be able to accept **cash and change**, as well as **university cash cards**, which require a **PIN** entry when used.
2. The machine will **sell food and drink** items just like other vending machines.
3. The machine will have a **unique ID and a location system** that will allow it to know what building it is in.
4. It will also have an advanced **theft deterrent and student protection system** that will use a **camera** and record the **image of the student** for all **cash card purchases** and record an image if the machine is being **shaken or tampered** with which is detected by an **accelerometer sensor**. During a tamper event, the machine will **call campus security** and provide **site ID** of the machine and the **photo of the perpetrators**.
5. The vending machine will also have a **built-in LED screen** that will provide several features, including:
    - advertise products based on a **sales trends** mined from the **products database**
    - mention **upcoming general school events** such as drop date, registration, etc.
    - advertise **student-customized information** and products such as university bills due, bookstore sales, etc. when the student uses their **M-Card** to buy their product.
6. The system will also interface with a **remote distribution center**. This system will **collect information regarding current product levels and buying trends** that will assist in **coordinating machine restocking**.

This page is intentionally left Blank.

Zachman Framework

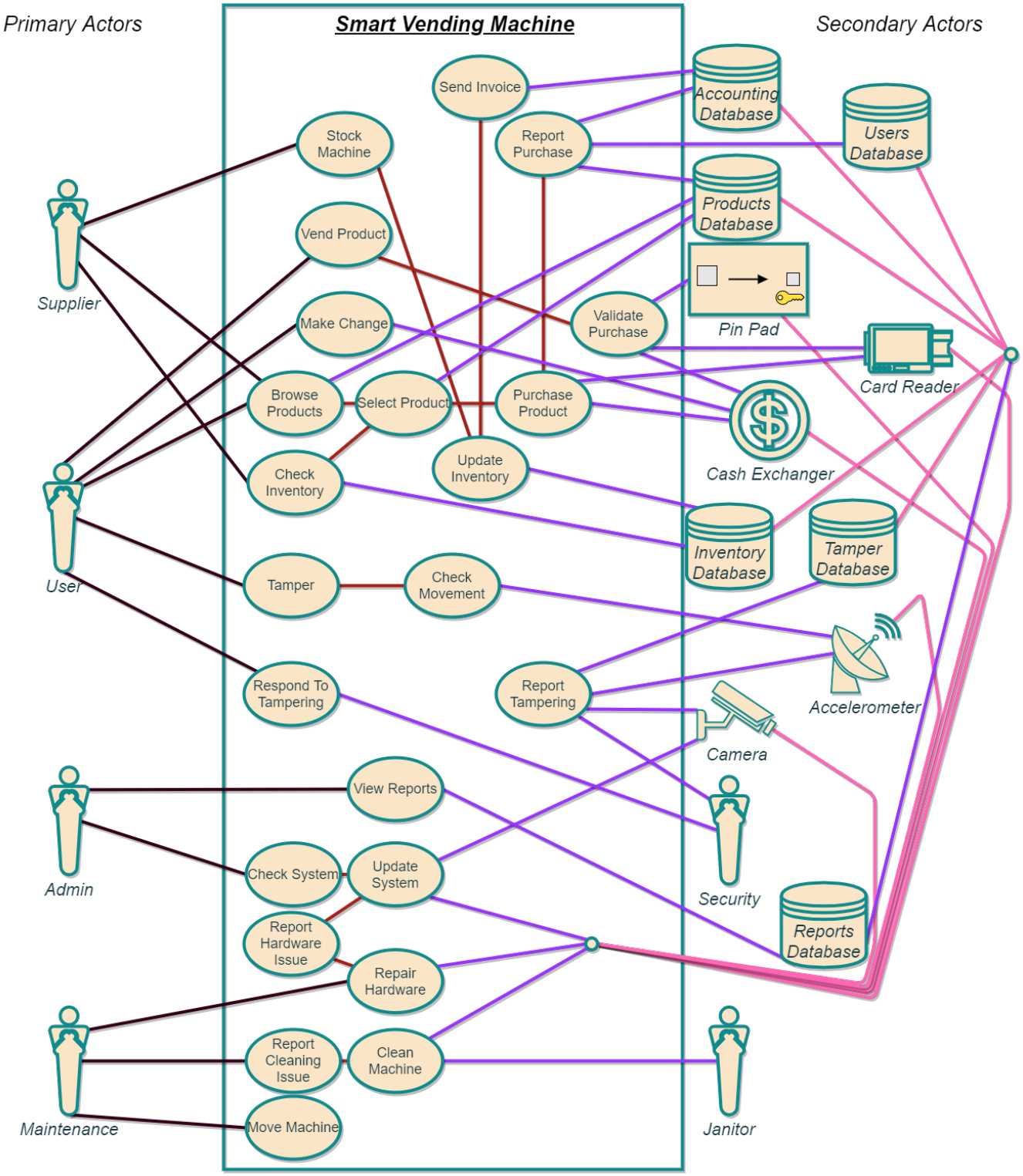
Pg. 1	*Planner* **Scope** (contextual perspective)
What **Data** (entities)	- Vending and sale of food and drink - Security threat detection - Verification and security imaging - School and product advertising - Student catered advertising - Device ID
How **Function** (activities)	- LED screen display [interface to facilitate interaction] - Cash and change exchanger [interface to facilitate cash payment] - University cash card reader [interface to facilitate card payment] - Pin pad [interface to facilitate pin verification and product selection] - Internet Connection [facilitate general information transfer] - Database Connections [facilitate specific data transfer on individual students, listed products and sales trends, campus schedule, and machine inventory] - Camera [device to facilitate verification and security imaging] - Accelerometer [device to facilitate security threat detection] - Cell servicing [facilitate transfer of automatic threat detection information (location, date/time, image) to campus security] - Email servicing [facilitate transfer of device inventory, machine status, product trends, user trends, net revenue, and threat history to admin]
Where **Network** (locations)	- College campus buildings. - Remote admin variable location.

Pg. 2	*Planner* **Scope** (contextual perspective)
Who **People**	- Campus students, faculty, staff, and guests [use] - Campus security [protect] - Campus IT [maintain software] - Campus maintenance [maintain hardware] - Campus janitorial [machine upkeep and cleaning] - Campus purchasing/accounting [machine cash flow and initial coordination for funding, supplying, and building] - Vending machine builders [outsourced company to implement the production of smart vending machines] - Food and drink suppliers [supply and stock products]
When **Time**	- Machine runs during university hours - Machine installation date - Machine initial testing [week following installation] - Machine launch date [1-week post installation] - Machine maintenance [every other month] - Machine cleaning [every week] - Software update/testing [quarterly] - Business continuation assessment [yearly]
Why **Motivation**	- Profit form food and drink sales - Provide safe no contact food and drink options to campus community - Continue campus integration of Web 3.0/IoT - Keep students informed of upcoming campus events to increase community involvement - Keep students informed on account status to reduce payment defaults - Provide additional campus security - Campus job creation

This page is intentionally left Blank.

Use Cases

[Context Diagram]



[Supplier Use Case 1]

	Description
Use Case:	*Browse products database, inform admin of issues.*
Primary Actor:	Supplier.
Goal in Context:	To use the system to search the universities database and compare with the marketing database. Supplier will inform admin of issues and suggest new products.
Preconditions:	Vending machines have been set up with internet access and the product catalog of items chosen by the university for sale.
Trigger:	Supplier notices some products have never been restocked and additionally has new products in their catalog.
Scenario:	1. Supplier: consults admin for access to products database. 2. Admin: grants supplier temporary database credentials. 3. Supplier: browse product database and compares with their database. 4. Supplier: takes notes of sales trends and compares with their sales to the university. 5. Supplier: notes one university product has never been restocked, notes this is not in the database. 6. Supplier: informs admin of product missing in database and recommends 2 new products. 7. Admin: orders 2 new products, fixes error in database and adds 2 new products as well.
Exceptions:	1. No products yet sold. 2. Machines not communicating with product database.
Priority:	Medium, manual product tracking possible.
When Available:	3rd iteration, before machine production.
Frequency of Use:	Monthly.
Channel to actor:	Remote database connection.

[Supplier Use Case 2]

	Description
Use Case:	*Stock machine, update database and send invoice.*
Primary Actor:	Supplier.
Goal in Context:	Stock all machines to full capacity, note products sold to university, send invoice for payment.
Preconditions:	Vending machines have been in service and selling a critical mass of products.
Trigger:	Vending machine runs out of stock of popular item.
Scenario:	1. User: approaches machine to by favorite soda, orange crush, and machine prompts out of stock. 2. User: upset, complains to food services. 3. Food Services: informs admin machine is out of orange crush soda and gives machine location. 4. Admin: cross references inventory reports, confirms machine is out as well as 3 other machines; contacts supplier to restock and sends inventory. 5. Supplier: brings delivery truck to school with needed inventory, carts restock from machine to machine and fills to the top. 6. Supplier: confirms inventory sold to university, leaves, and sends invoice for restock to university from office.
Exceptions:	1. Machine out of order for maintenance. 2. University on holiday and closed.
Priority:	Essential, at launch.
When Available:	First increment.
Frequency of Use:	Bi-weekly.
Channel to actor:	User: LCD-Screen. Supplier: Vending Machine, Email-comms.

[User Use Case 1]

	Description
Use Case:	*Buy soda drink with cash.*
Primary Actor:	User.
Goal in Context:	To use the system to buy a coca cola from a local university smart vending machine.
Preconditions:	Vending machine is stocked with cola, user has necessary funds to purchase (cash payment).
Trigger:	User is falling asleep in class and hates brown bean water, opts for tasty cola soda to get through class.
Scenario:	1. User: approaches kiosk and clicks on screen to end advertisements. 2. User: selects drink as product type. 3. User: browses sodas and selects coca cola. 4. User: reads product information and verifies he has the funds; selects purchase. 5. User: selects cash payment and inserts funds into cash exchanger. 6. Vending Machine: vends soda and returns change. 6. User: takes change and soda then leaves.
Exceptions:	1. Vending machine needs service. 2. Vending machine has no cash change. 3. Vending machine is out of coca cola. 4. No power to machine.
Priority:	Essential, must be implemented. (coca coal not)
When Available:	First increment.
Frequency of Use:	Many times, per day.
Channel to actor:	Vending Machine, LCD Display, Cash Exchanger.

[User Use Case 2]

	Description
Use Case:	*Shake machine to try and get free Fritos.*
Primary Actor:	User.
Goal in Context:	Force vending machine to drop free items via rigorous shake.
Preconditions:	Vending machine has working accelerometer, camera, and call functionality. Security available.
Trigger:	User has no funds and thinks can dubiously retrieve goods from vending machine.
Scenario:	1. User: hungry/thirsty approaches vending machine. 2. User: unwilling to pay for items shakes machine. 3. Accelerometer: triggers tamper waring. 4. Camera: images suspect, sends to security. 5. Vending Machine: calls security with location. 6. Security: responds to call. 7. Security: catches and changes perpetrator. 8. User: pays 100x item cost in fines.
Exceptions:	1. Security at doughnut shop. 2. Malfunction in accelerometer, imaging, or sending. 3. No power to machine.
Priority:	High, should be implemented within 1st week deployed.
When Available:	Second increment.
Frequency of Use:	Few times, per week.
Channel to actor:	Vending Machine, Security.

[Admin Use Case 1]

	Description
Use Case:	*View system reports.*
Primary Actor:	Admin.
Goal in Context:	Examine weekly reports generated by machine management services.
Preconditions:	Vending machines have working Wi-Fi and automatic email reporting is set up.
Trigger:	1 week passes and management system automatically compiles machine reports and forwards to admin.
Scenario:	1. Users: Frequent vending machine for 1 week. 2. Reports database: ques users report. 3. Reports database: ques products report. 4. Reports database: ques inventory report. 5. Reports database: ques accounting report. 6. Reports database: ques tamper report. 7. Reports database: compiles reports and forwards to admin vias email message. 8. Admin: reviews machine reports.
Exceptions:	1. Machine internet is down. 2. No power to machine. 3. Admin cannot access email or input wrong email.
Priority:	High, should be implemented within 1st week deployed.
When Available:	Second increment.
Frequency of Use:	Weekly.
Channel to actor:	Wi-Fi Module, Reports Database, Email.

[Admin Use Case 2]

	Description
Use Case:	*Report hardware issue to maintenance.*
Primary Actor:	Admin.
Goal in Context:	Ensure proper functionality of vending machines on campus.
Preconditions:	Minimum 1 vending machine has hardware issue, know maintenance provider available for machine.
Trigger:	1 week passes and management system automatically compiles machine reports and forwards to admin.
Scenario:	1. Reports database: compiles reports and forwards to admin vias email message. 2. Admin: reviews machine reports. 3. Admin: notices machine with MID XXX has bad status. 4. Admin: remote connects to machine XXX. 5. Admin: runs systems tests; notes bad component. 6. Admin: as component is hardware, informs maintenance of bad components and submits work order. 7. Maintenance: fixes bad component and reports work order. 8. Admin: updates accounting database and sets machine status for MID XXX to good.
Exceptions:	1. Machine internet is down. 2. No power to machine. 3. Admin cannot access email or input wrong email.
Priority:	High, should be implemented within 1st week deployed.
When Available:	Second increment.
Frequency of Use:	Couple times, per month.
Channel to actor:	Email, Wi-Fi Module, Vending Machine, Maintenance.

[Maintenance Use Case 1]

	Description
Use Case:	*Repair hardware*.
Primary Actor:	Maintenance.
Goal in Context:	Troubleshoot malfunctioning machine hardware, repair, and return to working order.
Preconditions:	Vending machine has bad hardware component and maintenance is notified.
Trigger:	Hardware component breaks, admin is notified and forwards to maintenance.
Scenario:	1. Admin: informed of malfunctioning machine via reports or direct communication. 2. Admin: attempts to troubleshoot issue remotely. 3. Admin: informs maintenance of bad components, specifics may be unknown, and submits work order. 7. Maintenance: arrives on site to troubleshoot bad component. 8. Maintenance: isolates problem and relays options to admin. 9. Admin: approves work. 10. Maintenance: fixes bad component and reports work order. 8. Admin: updates accounting database.
Exceptions:	1. No repair part available. 2. No repair funding available.
Priority:	High, should be implemented within 1st week deployed.
When Available:	Second increment.
Frequency of Use:	Couple times, per month.
Channel to actor:	Admin, Vending Machine.

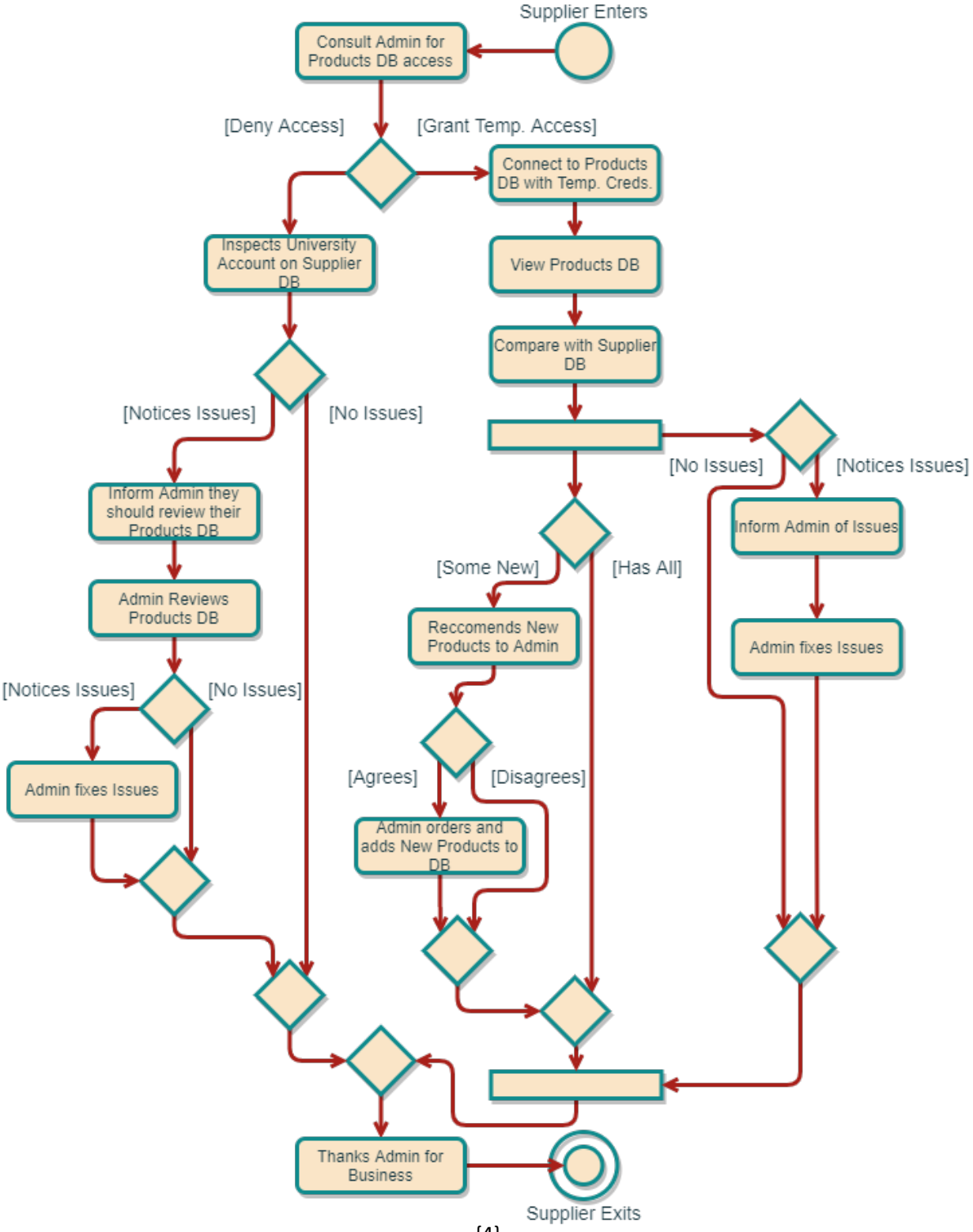
[Maintenance Use Case 2]

	Description
Use Case:	*Move vending machine location.*
Primary Actor:	Maintenance.
Goal in Context:	Move vending machine to new location.
Preconditions:	New location has internet access and power outlets.
Trigger:	Old building is being rebuilt and vending machine can no longer operate at the current location.
Scenario:	1. Security: informs admin building XXX is going to be rebuilt 2. Admin: finds next best location for vending machine in building YYY. 3. Admin: ensures building has available power outlets and internet access. 4. Admin: forwards information that old building is going to be rebuilt to maintenance, submits work order to move to building YYY. 5. Maintenance: obtains security clearance, moves vending machine from XXX to YYY, completes installation, informs admin and returns work order. 6. Admin: Tests system functionality and updates machine location tag.
Exceptions:	1. Vending Machine is permanently installed. 2. Building repairs blocking exit route.
Priority:	Essential, must be implemented.
When Available:	First increment.
Frequency of Use:	Couple times, per year.
Channel to actor:	Admin, Vending Machine.

This page is intentionally left Blank.

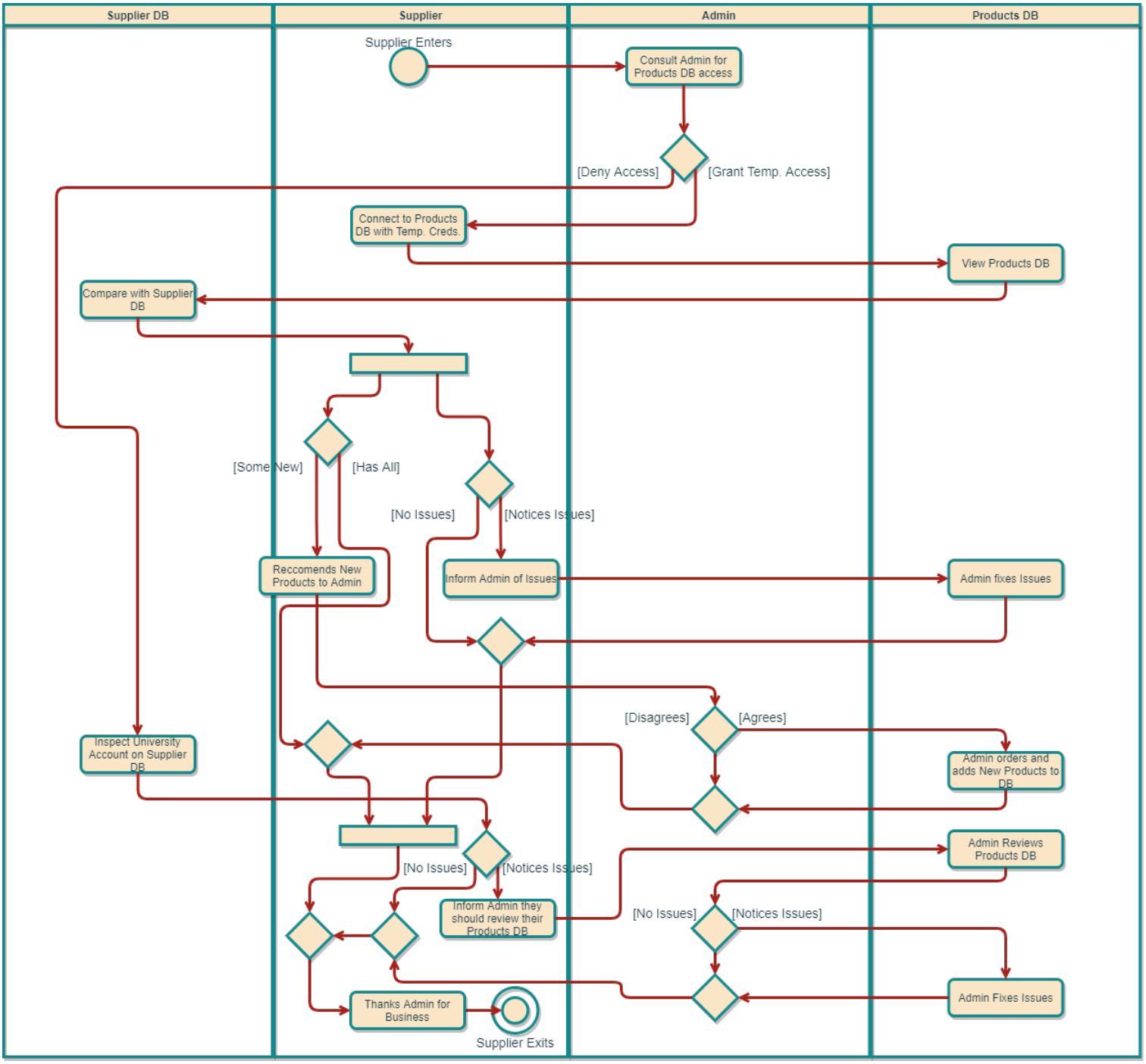
Activity and Swim Lane Diagrams

[Activity Diagram]



{4}

[Swim Lane Diagram]



This page is intentionally left Blank.

Candidate Classes and CRC Cards

[Stereotype Diagram]

1	**Actors**
	1.) User 2.) Admin 3.) Security 4.) Maintenance 5.) Supplier 6.) Janitor

2	**Devices**
	1.) Vending Machine 2.) LCD Screen 3.) Card Reader 4.) Cash Exchanger 5.) Pin Pad 6.) Wi-Fi Module 7.) Accelerometer 8.) Camera

3	**Interfaces**
	1.) Products 2.) Product 3.) Purchase 4.) Advertisement 5.) Lockout 6.) Service

4	**Databases**
	1.) Inventory Database 2.) Products Database 3.) Users Database 4.) Accounting Database 5.) Tamper Database 6.) Status Database 7.) Reports Database

[Actor CRC Cards]

Candidate Name	
+ User	
Purpose/Definition	
+ Purchase food or drink	
Responsibilities	**Collaborators**
Knows: + Cash Balance + Cash Card Balance + Machine Locations Does: + browse_products() + select_product(item) + pay(type) + verify(input_pin) + take(item)	+ Security + Vending Machine + LCD Screen + Card Reader + Cash Exchanger + Pin Pad + Camera + Products + Product + Purchase + Advertisment + Lockout + Service

Candidate Name	
+ Admin	
Purpose/Definition	
+ Maintain system software	
Responsibilities	**Collaborators**
Knows: + Login Credentials + Machine IDs & Locations + System Architecture Does: + login(MID, username, password) + update_credentials(username, password) + run_system_test() + update_test(test) + update_firmware(device) + run_query(query) + update_query(query) + update_interface(interface) + view_system_report(report) + lock_system() + unlock_system()	+ Security + Maintenance + Supplier + Janitor + Vending Machine + LCD Screen + Card Reader + Cash Exchanger + Pin Pad + Wi-Fi Module + Accelerometer + Camera + Products + Product + Purchase + Advertisment + Lockout + Servioce + Inventory Database + Products Database + Users Database + Accounting Database + Tamper Database + Status Database

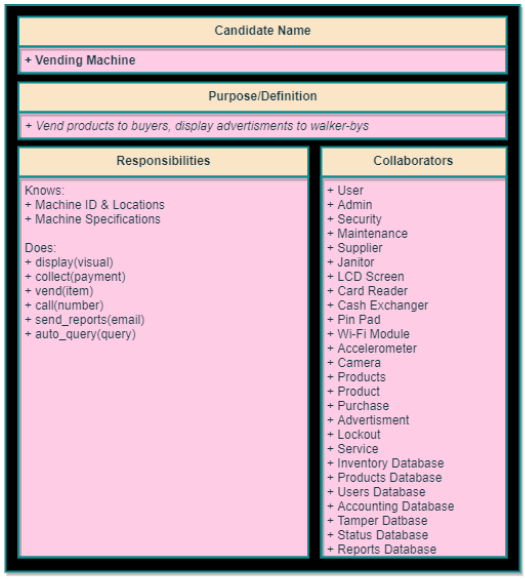
Candidate Name	
+ Security	
Purpose/Definition	
+ Respond to machine tampering events	
Responsibilities	**Collaborators**
Knows: + Badge # + Machine IDs & Locations + Suspect Photo Does: + respond(tamper_case) + make_report(tamper_case) + send_report(tamper_case)	+ User + Admin + Maintenance + Vending Machine + Lockout + Tamper History

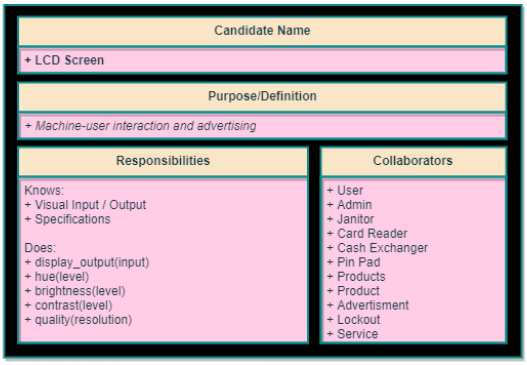
Candidate Name	
+ Maintenance	
Purpose/Definition	
+ Maintain system hardware	
Responsibilities	**Collaborators**
Knows: + Personell # + Machine IDs & Locations + Hardware Specifications Does: + lock_door(door) + unlock_door(door) + move(location) + replace(hardware) + fill_cash_exchanger() + troubleshoot(component)	+ Admin + Security + Janitor + Vending Machine + LCD Screen + Card Reader + Cash Exchanger + Pin Pad + Wi-Fi Module + Accelerometer + Camera + Service

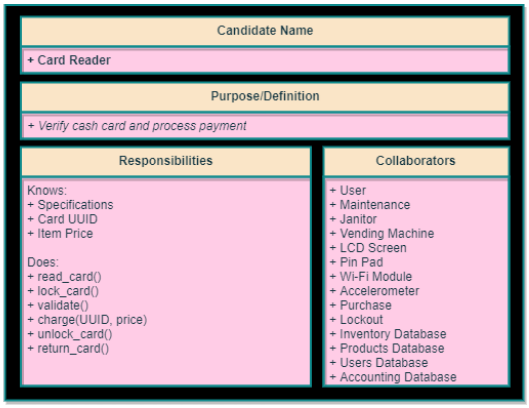
Candidate Name	
+ Supplier	
Purpose/Definition	
+ Stock machine	
Responsibilities	**Collaborators**
Knows: + Personell # + Machine IDs & Locations + Inventory + Product Details Does: + lock_door(door) + unlock_door(door) + refill(item) + run_inventory_update() + send_invoice() + send_details(products)	+ Vending Machine + Admin + Products + Product + Inventory Database + Products Database

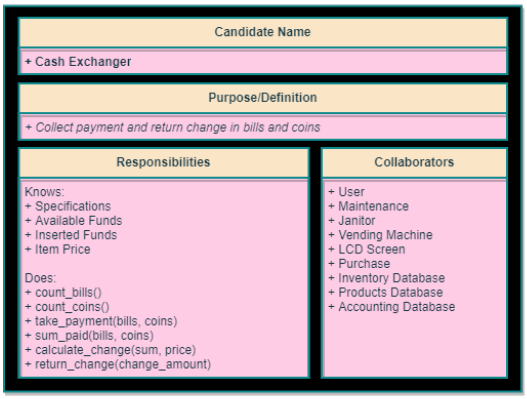
Candidate Name	
+ Janitor	
Purpose/Definition	
+ Clean machine	
Responsibilities	**Collaborators**
Knows: + Machine IDs & Locations Does: + clean() + report_damage(MID)	+ Admin + Maintenance + Vending Machine + LCD Screen + Card Reader + Cash Exchanger + Pin Pad + Camera + Service

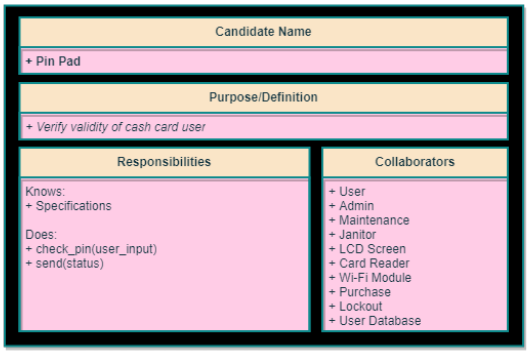
[Device CRC Cards]

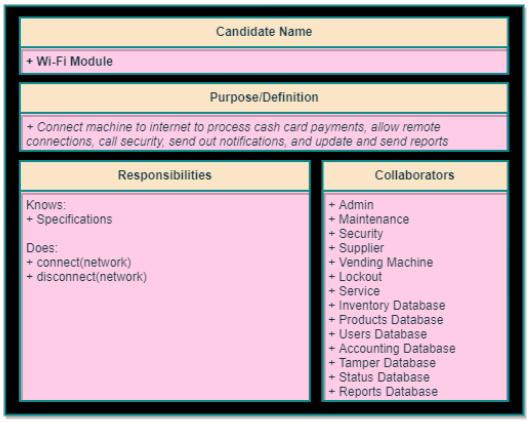


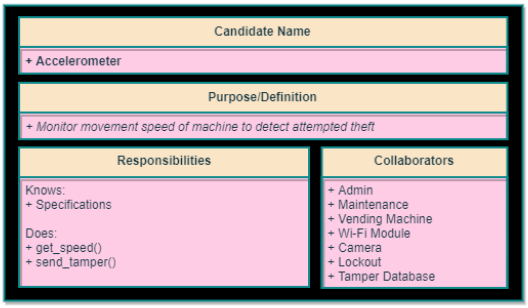


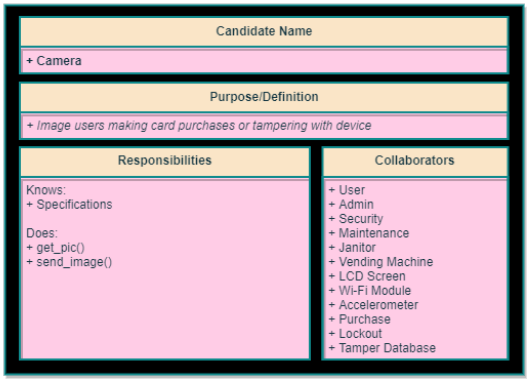












[Interface CRC Cards]

Candidate Name	
+ Products	
Purpose/Definition	
+ Interface providing user capability to browse and filter products	
Responsibilities	**Collaborators**
Knows: + Item List + Item Images + Item Prices + Item Attributes Does: + select(item) + select_type(type) + select_attribute(attribute) + set_min_price(price) + set_max_price(price) + view_best_sellers()	+ User + Admin + Vending Machine + LCD Screen + Wi-Fi Module + Product + Advertisement + Inventory Database + Products Database

Candidate Name	
+ Product	
Purpose/Definition	
+ Interface providing detailed product descriptions, pricing, ranking, and availability	
Responsibilities	**Collaborators**
Knows: + Name + Description + Images + Nutrition Facts + Price + Item Code + Item Inventory + Product Rank Does: + deselect() + purchase(item)	+ User + Admin + Vending Machine + LCD Screen + Wi-Fi Module + Products + Purchase + Advertisement + Inventory Database + Products Database

Candidate Name	
+ Purchase	
Purpose/Definition	
+ Interface providing user capability to purchase a product	
Responsibilities	**Collaborators**
Knows: + Item Code + Item Price + Item Inventory Does: + select_payment(type) + query_payment(type, price) + confirm_payment(choice)' + take_pic()	+ User + Admin + Vending Machine + LCD Screen + Card Reader + Cash Exchanger + Pin Pad + Wi-Fi Module + Camera + Product + Advertisement + Inventory Database + Products Database + Users Database + Accounting Database

Candidate Name	
+ Advertisment	
Purpose/Definition	
+ Interface used to display user advertisments	
Responsibilities	**Collaborators**
Knows: + Card UUID + Product Trends + University Calander Does: + student_ad(UUID) + university_ad(week) + product_ad(product)	+ User + Admin + Vending Machine + LCD Screen + Card Reader + Wi-Fi Module + Inventory Database + Products Database + Users Database

Candidate Name	
+ Lockout	
Purpose/Definition	
+ Interface used to display recent tamper incident and collect witness reports	
Responsibilities	**Collaborators**
Knows: + Incident Time + Machine Movement Does: + take_pic() + call_security() + lock_card() + report(suspect, witness) + send_witness_report() + timeout()	+ User + Admin + Security + Vending Machine + LCD Screen + Card Reader + Wi-Fi Module + Accelerometer + Camera + Tamper Database

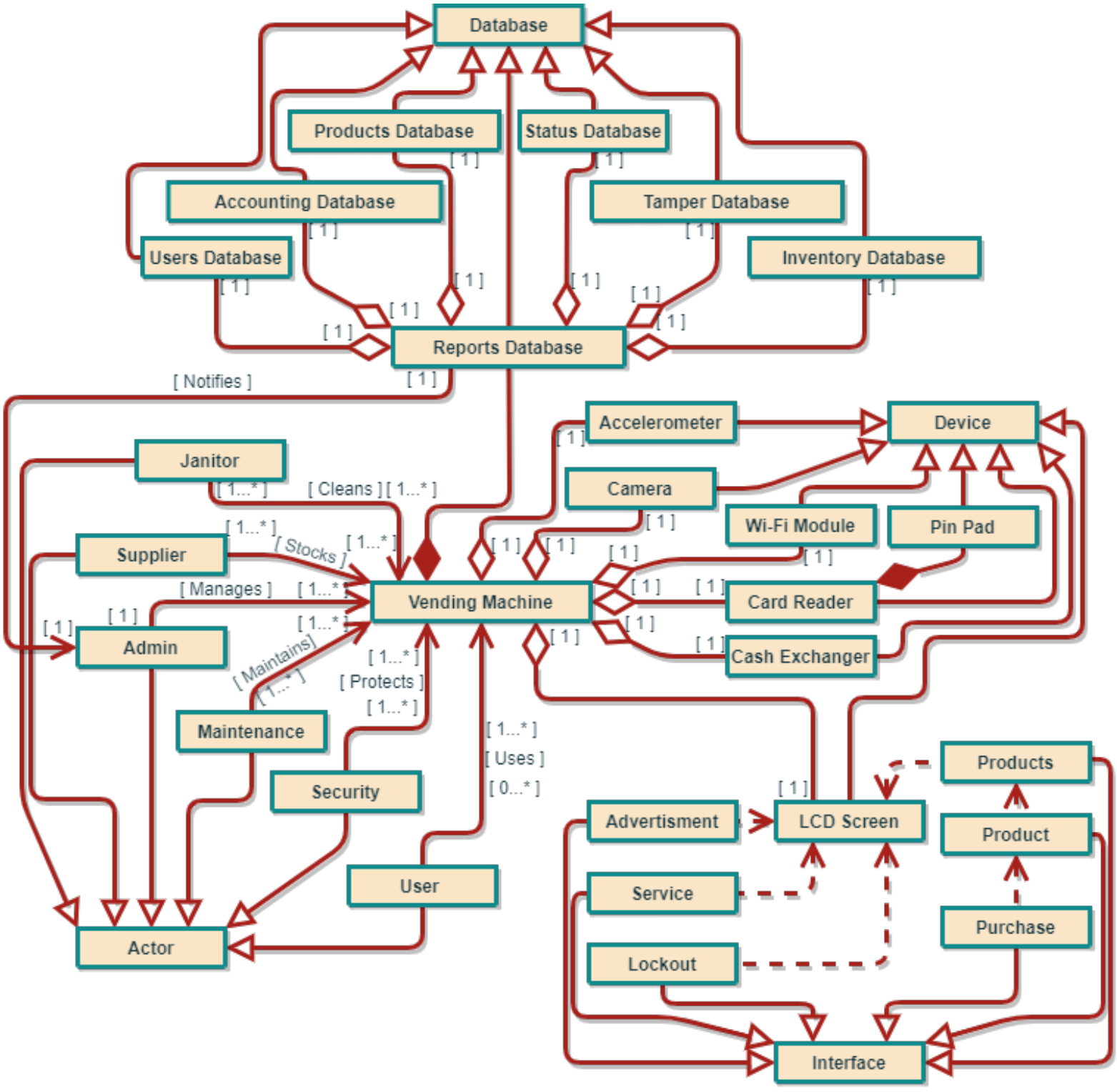
Candidate Name	
+ Service	
Purpose/Definition	
+ Interface used to display that system is down due to service needs	
Responsibilities	**Collaborators**
Knows: + Machine Status Does: + lock() + notify(status)	+ User + Admin + Maintenance + Vending Macing + LCD Screen + Wi-Fi Module + Status Database

[Database CRC Cards]

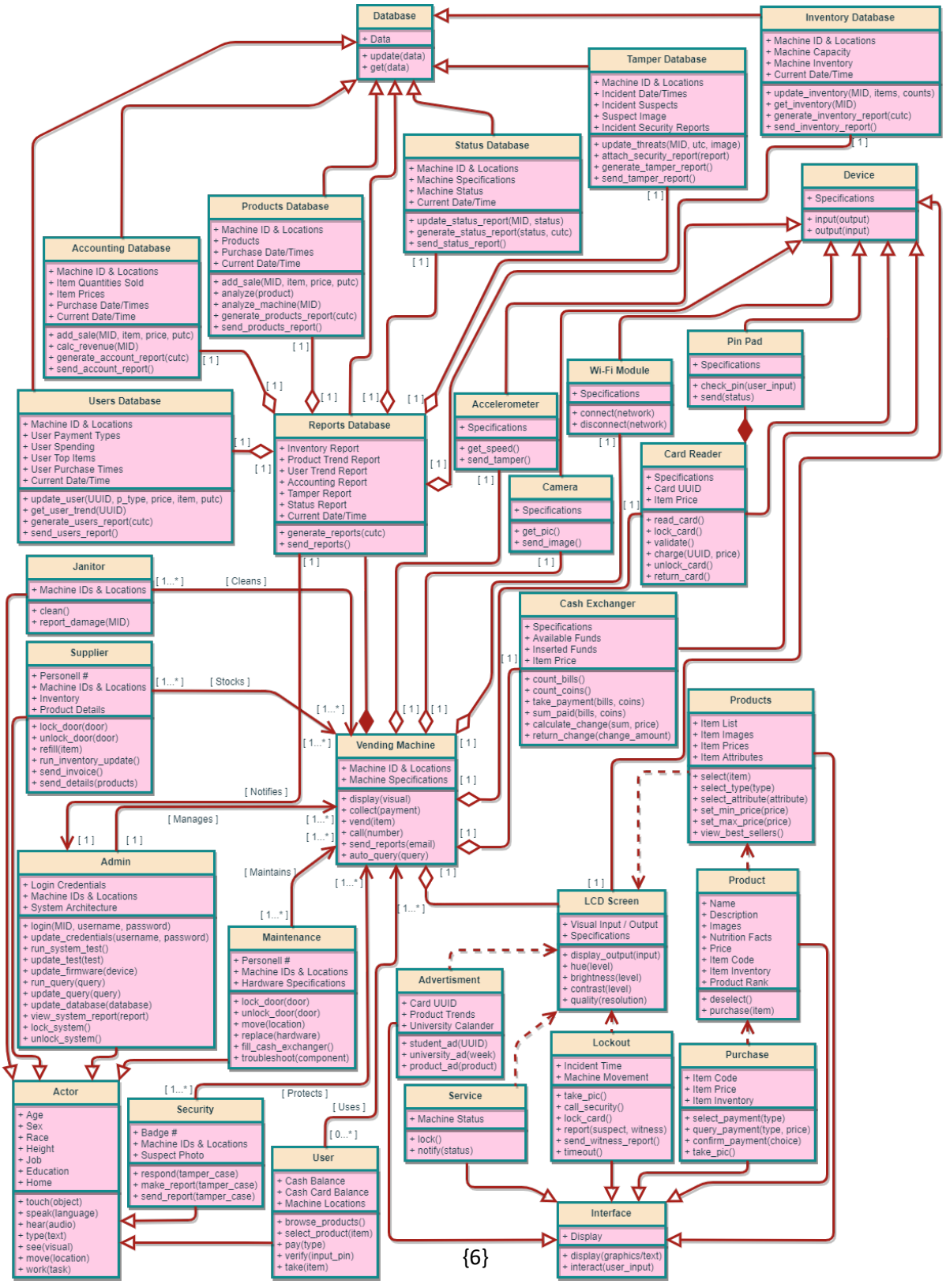
This page is intentionally left Blank.

Class Relationships

[Simplified]



[Detailed]

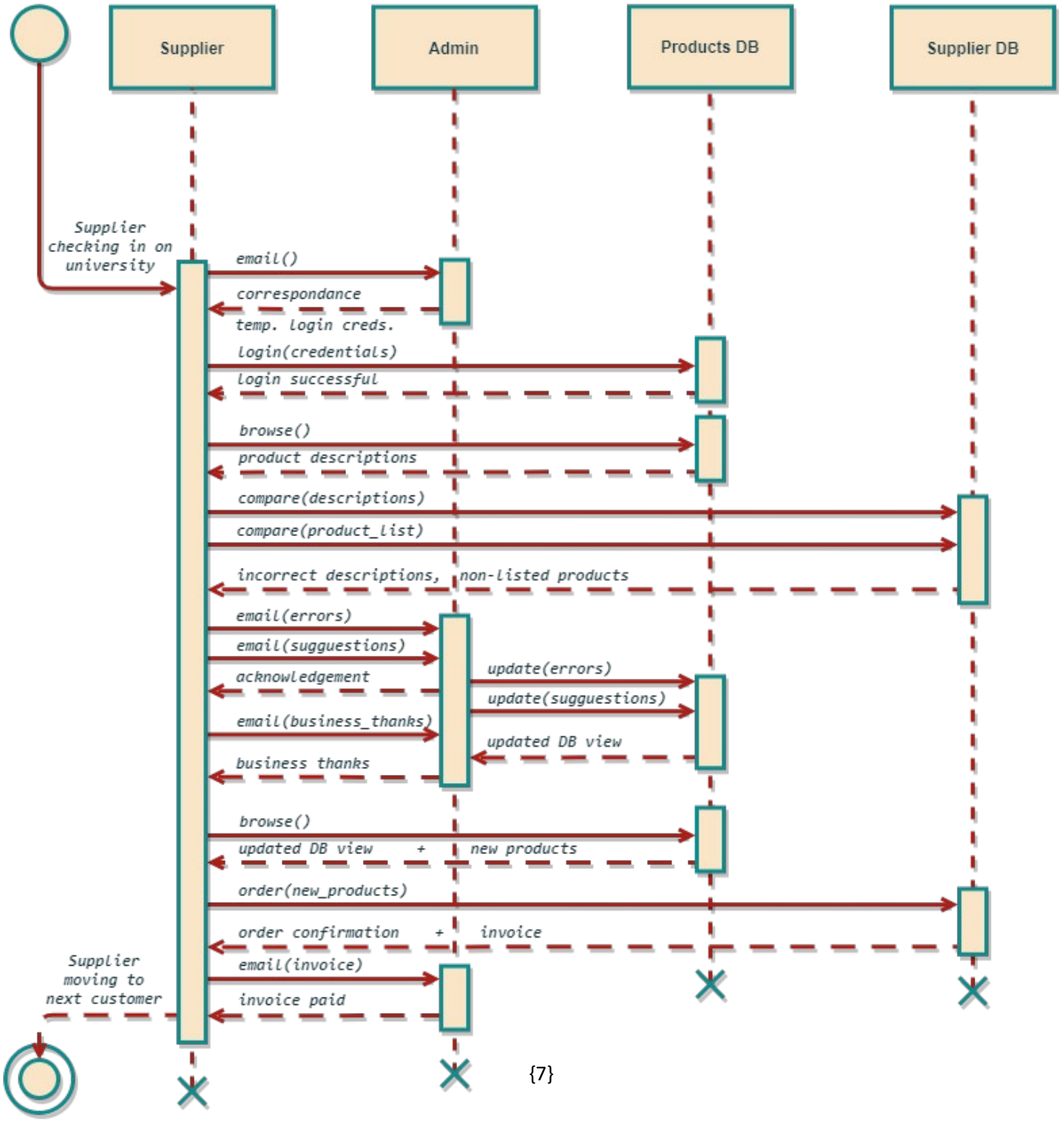

Database
+ Data
+ update(data)
+ get(data)
Inventory Database
+ Machine ID & Locations
+ Machine Capacity
+ Machine Inventory
+ Current Date/Time
+ update_inventory(MID, items, counts)
+ get_inventory(MID)
+ generate_inventory_report(cutc)
+ send_inventory_report()
Tamper Database
+ Machine ID & Locations
+ Incident Date/Times
+ Incident Suspects
+ Suspect Image
+ Incident Security Reports
+ update_threats(MID, utc, image)
+ attach_security_report(report)
+ generate_tamper_report()
+ send_tamper_report()
Status Database
+ Machine ID & Locations
+ Machine Specifications
+ Machine Status
+ Current Date/Time
+ update_status_report(MID, status)
+ generate_status_report(status, cutc)
+ send_status_report()
Device
+ Specifications
+ input(output)
+ output(input)
Products Database
+ Machine ID & Locations
+ Products
+ Purchase Date/Times
+ Current Date/Time
+ add_sale(MID, item, price, putc)
+ analyze(product)
+ analyze_machine(MID)
+ generate_products_report(cutc)
+ send_products_report()
Accounting Database
+ Machine ID & Locations
+ Item Quantities Sold
+ Item Prices
+ Purchase Date/Times
+ Current Date/Time
+ add_sale(MID, item, price, putc)
+ calc_revenue(MID)
+ generate_account_report(cutc)
+ send_account_report()
Pin Pad
+ Specifications
+ check_pin(user_input)
+ send(status)
Wi-Fi Module
+ Specifications
+ connect(network)
+ disconnect(network)
Users Database
+ Machine ID & Locations
+ User Payment Types
+ User Spending
+ User Top Items
+ User Purchase Times
+ Current Date/Time
+ update_user(UUID, p_type, price, item, putc)
+ get_user_trend(UUID)
+ generate_users_report(cutc)
+ send_users_report()
Reports Database
+ Inventory Report
+ Product Trend Report
+ User Trend Report
+ Accounting Report
+ Tamper Report
+ Status Report
+ Current Date/Time
+ generate_reports(cutc)
+ send_reports()
Accelerometer
+ Specifications
+ get_speed()
+ send_tamper()
Card Reader
+ Specifications
+ Card UUID
+ Item Price
+ read_card()
+ lock_card()
+ validate()
+ charge(UUID, price)
+ unlock_card()
+ return_card()
Camera
+ Specifications
+ get_pic()
+ send_image()
Janitor
+ Machine IDs & Locations
+ clean()
+ report_damage(MID)
Cash Exchanger
+ Specifications
+ Available Funds
+ Inserted Funds
+ Item Price
+ count_bills()
+ count_coins()
+ take_payment(bills, coins)
+ sum_paid(bills, coins)
+ calculate_change(sum, price)
+ return_change(change_amount)
Supplier
+ Personell #
+ Machine IDs & Locations
+ Inventory
+ Product Details
+ lock_door(door)
+ unlock_door(door)
+ refill(item)
+ run_inventory_update()
+ send_invoice()
+ send_details(products)
Products
+ Item List
+ Item Images
+ Item Prices
+ Item Attributes
+ select(item)
+ select_type(type)
+ select_attribute(attribute)
+ set_min_price(price)
+ set_max_price(price)
+ view_best_sellers()
Vending Machine
+ Machine ID & Locations
+ Machine Specifications
+ display(visual)
+ collect(payment)
+ vend(item)
+ call(number)
+ send_reports(email)
+ auto_query(query)
Admin
+ Login Credentials
+ Machine IDs & Locations
+ System Architecture
+ login(MID, username, password)
+ update_credentials(username, password)
+ run_system_test()
+ update_test(test)
+ update_firmware(device)
+ run_query(query)
+ update_query(query)
+ update_database(database)
+ view_system_report(report)
+ lock_system()
+ unlock_system()
LCD Screen
+ Visual Input / Output
+ Specifications
+ display_output(input)
+ hue(level)
+ brightness(level)
+ contrast(level)
+ quality(resolution)
Product
+ Name
+ Description
+ Images
+ Nutrition Facts
+ Price
+ Item Code
+ Item Inventory
+ Product Rank
+ deselect()
+ purchase(item)
Maintenance
+ Personell #
+ Machine IDs & Locations
+ Hardware Specifications
+ lock_door(door)
+ unlock_door(door)
+ move(location)
+ replace(hardware)
+ fill_cash_exchanger()
+ troubleshoot(component)
Advertisment
+ Card UUID
+ Product Trends
+ University Calander
+ student_ad(UUID)
+ university_ad(week)
+ product_ad(product)
Lockout
+ Incident Time
+ Machine Movement
+ take_pic()
+ call_security()
+ lock_card()
+ report(suspect, witness)
+ send_witness_report()
+ timeout()
Purchase
+ Item Code
+ Item Price
+ Item Inventory
+ select_payment(type)
+ query_payment(type, price)
+ confirm_payment(choice)
+ take_pic()
Service
+ Machine Status
+ lock()
+ notify(status)
Actor
+ Age
+ Sex
+ Race
+ Height
+ Job
+ Education
+ Home
+ touch(object)
+ speak(language)
+ hear(audio)
+ type(text)
+ see(visual)
+ move(location)
+ work(task)
Security
+ Badge #
+ Machine IDs & Locations
+ Suspect Photo
+ respond(tamper_case)
+ make_report(tamper_case)
+ send_report(tamper_case)
User
+ Cash Balance
+ Cash Card Balance
+ Machine Locations
+ browse_products()
+ select_product(item)
+ pay(type)
+ verify(input_pin)
+ take(item)
Interface
+ Display
+ display(graphics/text)
+ interact(user_input)
[Cleans]
[Stocks]
[Notifies]
[Manages]
[Maintains]
[Protects]
[Uses]

This page is intentionally left Blank.

Sequence Diagram/Charts and State Diagrams

[Products Sequence Diagram]

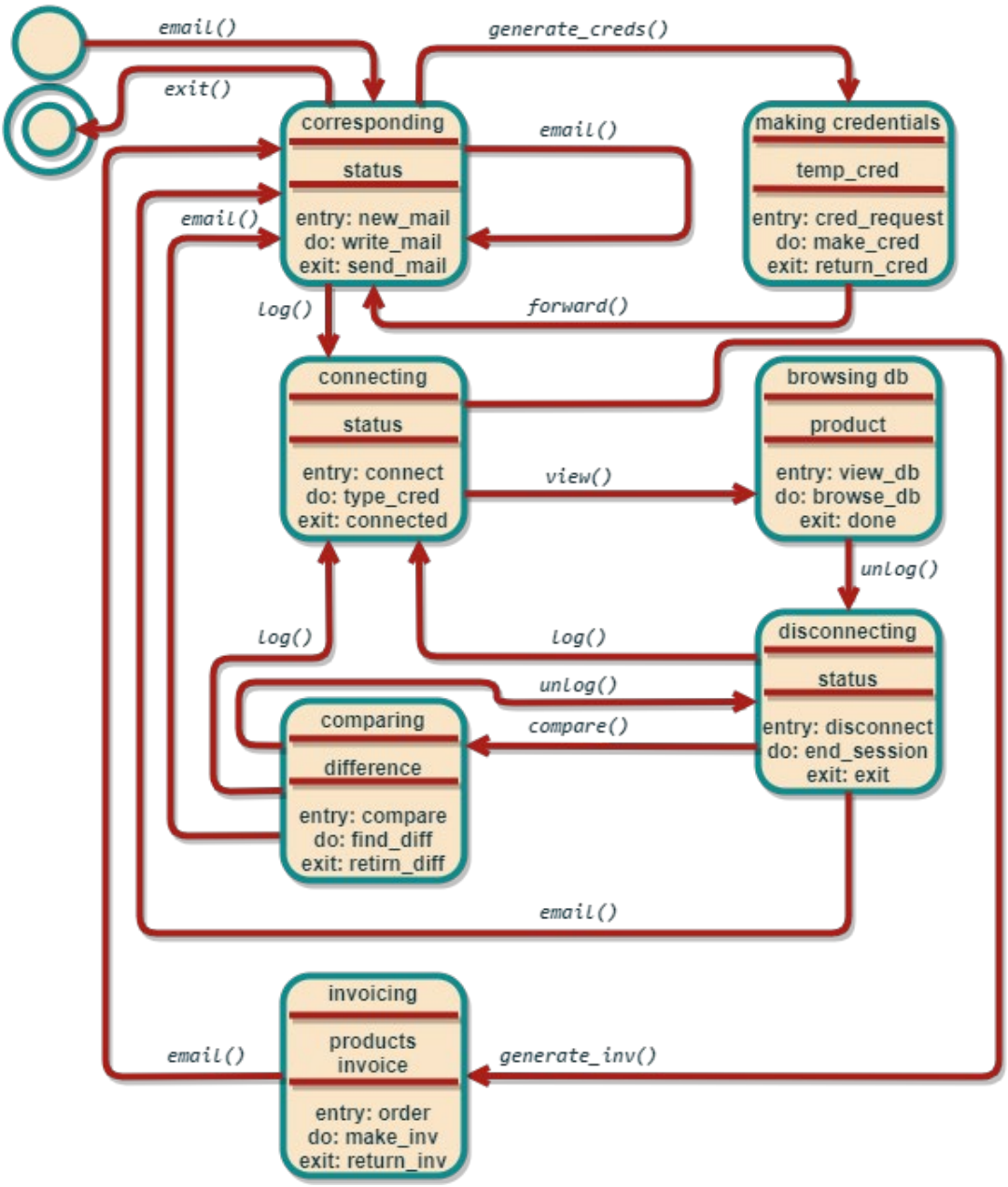
Supplier Browses Products DB, informs on issues.



{7}

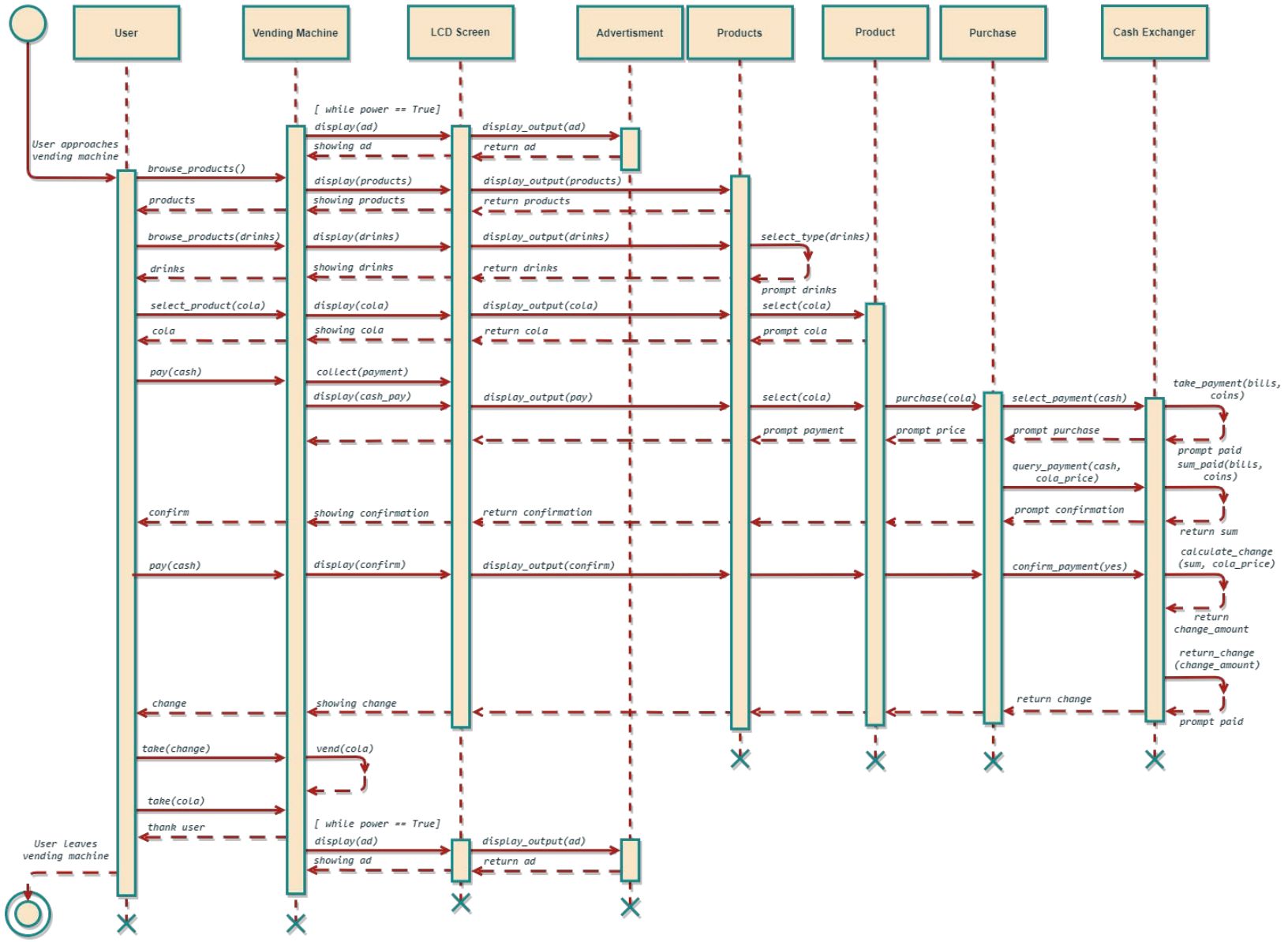
[Products State Diagram]

Supplier Browses Products DB, informs on issues.

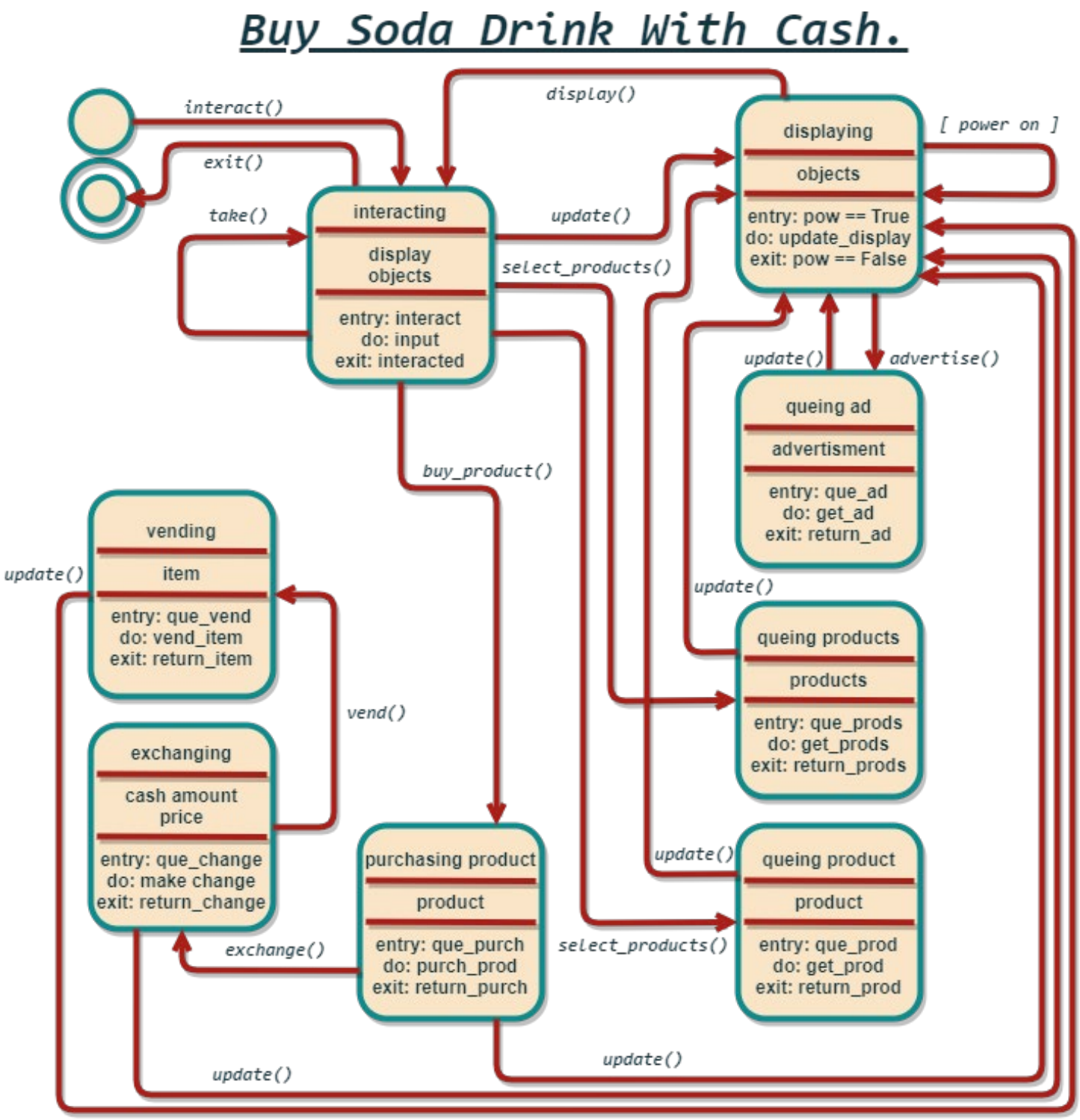


[Purchase Sequence Diagram]

Buy Soda Drink With Cash.



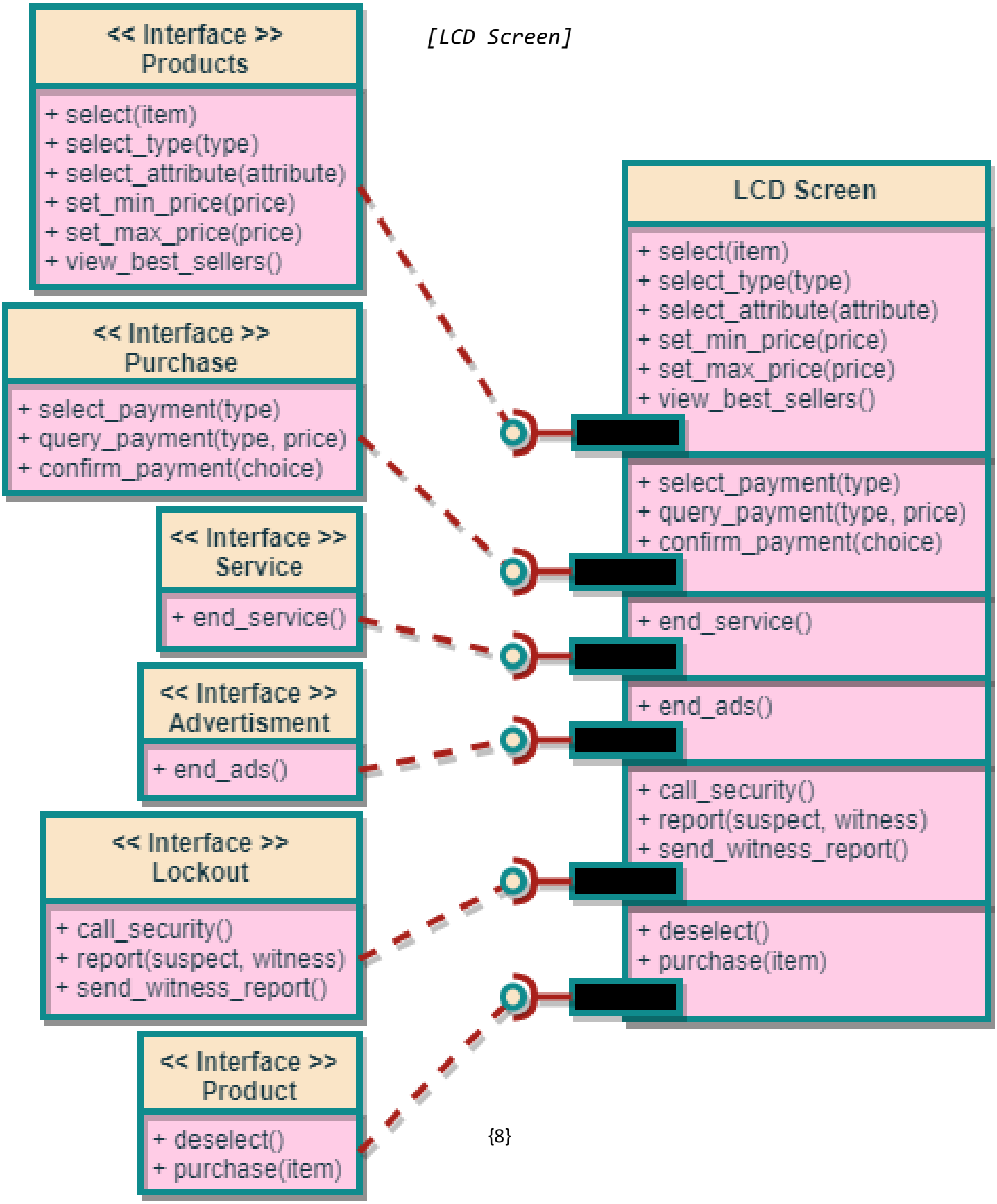
[Purchase State Diagram]



This page is intentionally left Blank.

Components

[LCD Screen]



{8}

This page is intentionally left Blank.

Testing

After conducting business meetings with the system architect and developer we have decided to outline critical testing needs while working additionally with an independent testing organization. Both the architect and developer have made clear to the business that they have limited resources to spend on this project and will be able to support the tester but not able to test in full before launch. As the project is also limited in funding, we have decided to take a black box approach to testing. Before development starts will we forward design plans to the agency to get a third-party opinion on the system's architecture. Within the scope of testing, we have isolated key functionality that needs to be rigorously tested before deployment: system interfacing, cash and cash card purchasing, report consolidation and automatic sending, individual database access, tamper notification and accelerometer thresholding, bandwidth, and security. Following initial black box testing we will have an idea as to what software aspects are faulty. With this new information we will start white box testing techniques through pair programming between the tester and developer. If we have an issue not easily dissolved, we will consult the architect to ensure it is not a design issue. Before forwarding software modules to the independent tester, we have asked the developer to do base-line testing to ensure communication channels can be reached and data objects are readable. We would like the independent tester to design a set of unit tests based on the use cases we provide; at minimum, the software needs to be able to carry out the use cases. We will also provide the tester with required specifications for the system. The unit will have a bandwidth limit that must not be exceeded as well as time limits on state transfer. The team will utilize code stepping and break points such to make the testing process easier and ensure isolation of problematic blocks. We expect the software development to occur over a 3-month time. Within this time the tester will have weekly meetings scheduled with the developer and every third meeting additionally with the architect. The first and last meeting will also encapsulate the technical project manager to lay the foreground and address what did and did not work throughout the testing process. If the system does not have a functional bug, but rather a specification bug we will forward to the architect to discuss a prompter meeting on ideas of algorithm optimization and design simplifications, KISS!

While in the software development phase the developer is to work with the sys admin to discuss the best possible formatting for reports and remote access. The tester will verify the formatting with the admin and ensure the system still operates within its specifications. Given typical operation is working we will ask the admin to start playing with the system and getting a feel. As a result, the admin will become familiarized with bug handling and act as an alpha tester for the project.

Once we have obtained system functionality it is time to start recovery testing. In this phase of testing, we will go over the common fault cases and ensure system recovery: power outage, internet outage, firmware rollback, hardware disconnect/damage, and os crash. The recovery testing will be handled with both the developer and the unit maintainer. We will obtain faulty hardware and firmware from the component manufactures. The technician will install faulty components and introduce errors while the developer creates workarounds, and the tester verifies proper recovery. The system should utilize checkpoints and store previous drivers on the onboard memory unit.

At this point we will turn a full unit over to the testing agency to conduct their pen testing. The pen testing should be rigorous and encapsulate a variety of attacks: software, hardware, internet, and false verification. Additionally, we will provide the organization with the necessary accessories: remote access credentials, valid cash card, and local configuration requirements. It will be the goal of the testers to inform the university of the extent to which the system can be penetrated. We acknowledge that there is many ways to get into systems but want to ensure that doing so must take not only a high technical expertise but also a large degree of time expenditure. It is our requirement that penetration must require a large sum of time in or near the system such to warrant suspicious actors. Granted there is easy penetration we will work with the respective parties to ensure a patch is put in place. If the breach is due to hardware, we will consider possible substitutes with a more rigorous security system.

At this point in the testing plan we have ensure that the system can pass our use cases, meet specifications, and is not easily penetrated. If for any reason these tests are not passed it is essential to readdress at this point before we continue to the next phase. If no objections are made, we will now move to beta testing. On campus we have a multitude of university buildings well suited for such a smart vending machine. It is to our great benefit that we have

a large student and faculty body that we can utilize as a testing group. The first unit will be installed outside the CSC department staff offices. We will forward each faculty member a university cash card with $15 loaded and ask for them to take the opportunity to test out the new addition. During this timeframe we will ask for feedback from the end users on the system UI and UX. The developer will work with faculty to implement those changes agreed on as suitable by the project manager. We will implement and then run the experiment again throughout a second beta group of university employees located nearby to the CSC department. Again, we will ask for feedback and implement changes. After any changes made, we will have the sys admin run the unit test scripts provided by the independent testing agency. It is at this point we will move into phase 2 of beta testing.

In phase 2 beta testing we will install 2 units with the UPAV building, one of each end of the cafeteria. Students will be able to use their M-Card and cash to make product purchases from the new smart vending machine. On the machines we will have a scannable QR code linked to a forum to report errors to the sys admin. Given there is any common errors found we will forward the description to the developer to debug. This phase of beta testing will run for 2 university semesters and have an additional QR code for user suggestions. Again the developer will work with the suggestion to improve UI/UX, the sys admin will be asked to test the functionality. As the university now has 3 units in place, we will A/B test the changes to see if there is positive feedback within the university community. After phase 2 of beta testing has completed the project manager has put in place a plan to outfit all university buildings with the new smart vending machines.

These new machines will be ordered in bulk and the university will purchase a 10 for a total of 13 machines. It is the university desire to increase the community satisfaction and ease of access. As a result, the machines will have a scannable QR code for product suggestions.

Due to rapid technological advancements the university will stay in touch with the testing organization to conduct yearly pen tests. New vulnerabilities will be made aware to the sys admin who will now be responsible for consulting on, or individually pursuing, the methods available to combat security vulnerabilities and ensure the safety and privacy of university community members.