Presenter, company name and addressTitle and subtitleVersion number and dateDecorative sidebar

Table of Contents

[**Project Description**](#_heading=h.vjd678ec33f3) **2**

[Overview (ABET-2)](#_heading=h.30j0zll) 2

[Global Trends (EM@FSE-e)](#_heading=h.4kuqzun7ezs9) 2

[Market Analysis (EM@FSE-k)](#_heading=h.55n1k9y1dy6p) 2

[Security Considerations (SER-2)](#_heading=h.5xwvyfnqriee) 3

[Key Requirements (SER-2)](#_heading=h.1fob9te) 3

[Deliverables (SER-1)](#_heading=h.3znysh7) 4

[Acronyms and Abbreviations (ABET-3):](#_heading=h.2et92p0) 4

[**Design and Architecture**](#_heading=h.6947h1kc51rt) **5**

[Design Description (ABET-1, ABET-2)](#_heading=h.bb2hp88o243u) 5

[Alternate Design Possibilities (EM@FSE-b)](#_heading=h.q1elxyg8t9ue) 7

[**Implementation Strategy**](#_heading=h.3dy6vkm) **10**

[High-level Work Breakdown Structure (SER-1)](#_heading=h.1t3h5sf) 10

[Schedule / Timeline (SER-1):](#_heading=h.4d34og8) 11

[Required Hardware (SER-1, EM@FSE-o):](#_heading=h.2s8eyo1) 12

[Third-party content (SER-1, EM@FSE-o):](#_heading=h.17dp8vu) 13

[Quality (SER-2):](#_heading=h.3rdcrjn) 13

[References/Sources of Information (EM@FSE-q)](#_heading=h.e0klir9fr37l) 14

[Scalability (EM@FSE-J)](#_heading=h.7dkf8owe12hy) 14

[Other Special considerations (ABET-2)](#_heading=h.lnxbz9) 15

[**Process**](#_heading=h.35nkun2) **15**

[Process Description and Justification (SER-1)](#_heading=h.1ksv4uv) 15

[Tools (SER-1, EM@FSE-O)](#_heading=h.44sinio) 15

[Roles and Responsibilities (SER-1):](#_heading=h.2jxsxqh) 15

[Location of Project Artifacts (SER-1):](#_heading=h.z337ya) 16

[Sponsor communications (ABER-3)](#_heading=h.3j2qqm3) 16

[**Risk management**](#_heading=h.1y810tw) **16**

[Identified Potential Risks (SER-2)](#_heading=h.4i7ojhp) 16

[Mitigation Strategies (SER-2):](#_heading=h.2xcytpi) 18

[**References**](#_heading=h.9t9pbnezgxey) **20**

# Project Description

## Overview (ABET-2)

Speedy Snacks sells snack deliveries from brick-and-mortar stores through multiple online delivery apps. They have no snacks of their own and no delivery people of their own. The business plan includes partnering with mobile app food delivery companies like Uber Eats, Skip the Dishes, and DoorDash, and with restaurants and convenience stores. These companies handle the payment processing and delivery and are referred to as fulfillment partners.

Currently, delivery companies provide a tablet to the fulfillment partners. These tablets receive orders placed by customers ordering food from their mobile devices. If the item requested is in stock the order is accepted by a store attendant and a notification is sent out that the order is available and ready for pick up. A delivery company then sends a driver to pick up and deliver the order.

The problem Speedy Snacks is looking to address is that each delivery company they partner with provides a separate device. Speedy Snacks consolidates these to one tablet with one app that communicates with and accepts customer orders from multiple delivery apps. They would like a tool to automate menu uploads from stores because many of them, particularly convenience stores, will have menus with hundreds of options. By streamlining the store owner’s experience of accepting orders from multiple delivery apps and managing their menu, Speedy Snacks will eliminate confusion and present a more appealing service to fulfillment partners.

The final deliverable to Speedy Snacks should have an intuitive UI, designed with Webflow or a similar tool. A cross-platform front end for Android and iOS tablets will be developed with Flutter that communicates with a back-end API hosted in AWS. The back end will be developed on Django’s Python framework along with MySQL for persistent storage.

The primary target market for this business is convenience stores who currently do not use any technology for marketing or sales and who are looking to expand their business online. These clients will appreciate the instant access this service provides to delivery app customers without the headache of selecting and setting up the apps themselves.

## Global Trends (EM@FSE-e)

This project fits into the context of Big Data. Speedy Snacks is reaching out to convenience stores who currently do not have an online presence and helping them reach an underserved online customer base, according to the sponsor’s market research. The result will enable Speedy Snacks to compile and analyze customer purchasing data and report the results to each fulfillment partner. This data can be used to predict trends and inform future inventory decisions.

It also promotes the Sustainability of cities by facilitating transportation services that already exist but have not reached their market potential. Logistics reform may not be a shiny, exciting area of development but it is still important for the viability of urban living. Convenient snack deliveries may not be as revolutionary as ice deliveries in the early 20th century, but it makes cities slightly more pleasant to live in and that exerts a small but real pressure on demographic trends.

## Market Analysis (EM@FSE-k)

The market for this product includes convenience store owners who currently do not have an online presence and are looking to expand their business. There are several solutions already in the market. Two of them are [Cuboh](http://www.cuboh.com) and [Deliverect](http://www.deliverect.com). Cuboh defeats “tablet hell” by consolidating all orders from delivery partners into a single tablet. Fulfillment partners can manage all of their menus from a central hub. Cuboh claims to onboard new fulfillment partners in as little as one to ten days depending on the complexity of their menus. Deliverect similarly integrates incoming orders in a management application. Their product focuses on tools for managing inventory and item availability.

Our project provides significantly higher value than the existing solutions by focusing on the niche market of just snack deliveries, which opens up the possibility of ordering snacks from convenience stores. Aside from offering hundreds or thousands of snack options, convenience store prices are significantly lower than restaurants for the same item. For example, a bag of three chocolate chip cookies from Subway costs $1.70, whereas a dozen cookies from Kroger costs $3.99. It also offers fulfillment partners a custom-built system to automate the process of adding their inventory. This means one to two-day onboarding and much less hassle to get up and running, as opposed to the one-size-fits-all approach provided by other solutions.

## Security Considerations (SER-2)

The potential downside of data breaches is very low because the customer data is restricted to non-sensitive things. Because payments and deliveries are handled by outside systems, sensitive information like the customer’s credit card number, address, and phone number are not required.

The application will be downloaded from the Android and iOS app stores but there will be no option to self-register on the home page, user accounts will instead be created by Speedy Snacks staff during onboarding. Passwords will be randomly generated and access to the application will be restricted to business partners who can log in. Communication from the front end to the back end will need to be encrypted. Back-end data will also be encrypted. Back-end data will be restricted to system administrators but fulfillment partners will have access to their own data. A log of all activity per user account will be stored as a history.

These precautions cover all the potential security breaches of which we are aware. There are no specific security risks remaining that could be mitigated to our knowledge, and as explained above the downside of the security risks we are not aware of is very low.

## Key Requirements (SER-2)

The application must have an intuitive GUI that works on both iOS and Android operating systems. Menu entry must support batch uploads and the menu must display pictures of each product available. It must interact with multiple third-party APIs to receive orders and update the status of received orders. The back-end server must support at least 400 stores handling 1,000 transactions per month.

System administrators at Speedy Snacks must be able to create accounts for fulfillment partners who will then each be able to log into their account by entering an email and password. Users must have the option to change their passwords by clicking on “I forgot my password”. Managers must be able to add, edit, and delete employee profiles.

The front end must allow fulfillment partners access to add, edit, and delete inventory items. Generated sales reports must be available to managers. Fulfillment partners must be able to contact system administrators through a contact form in the application. The system must be designed to restrict access to sensitive information such as customer data to minimize liability concerns.

## Deliverables (SER-1)

TABLE I

SUMMARY OF MAJOR PROJECT DELIVERABLES

| **Deliverable Number** | **Deliverable Name** | **Deliverable description** | **Status** | **Dependency** |
| --- | --- | --- | --- | --- |
| 1 | Project plan | Identify scope and timeline of the project. | In Progress | None |
| 2 | Wireframe designs | Design the layout of the various application screens. | Not Started | Deliverable 1 |
| 3 | Prototype GUI | Create a working login and landing page, and prototypes of all other pages. | Not Started | Deliverable 1, 2 |
| 4 | Front-end services | Prepare API calls so that front end functions with test data. | Not Started | Deliverable 1, 2, 3 |
| 5 | Design business logic and persistent storage | Design handling and return of API calls from in-house or third-party APIs. Design persistent storage of application data. | Not Started | Deliverable 1, 2, 3, 4 |
| 6 | Back-end services | Develop API using websockets with Python’s Django framework and MySQL. | Not Started | Deliverable 1, 2, 3, 4, 5 |
| 7 | QA Testing | Testing to validate software. (See subsection “Quality”.) | Not Started | Deliverable 1, 2, 3, 4, 5, 6 |

## Acronyms and Abbreviations (ABET-3):

TABLE II

ACRONYMS AND ABBREVIATIONS

| **Acronym** | **Description** |
| --- | --- |
| API | Application Programming Interface |
| QA | Quality Assurance |
| GUI | Graphical User Interface |
| IDE | Integrated Development Environment |
| UC | Use Case |
| AWS | Amazon Web Service |
| iOS | iPhone Operating System |
| RAM | Random Access Memory |
| LAN | Local Area Network |
| VS | Visual Studio |

# Design and Architecture

## Design Description (ABET-1, ABET-2)

The program as a whole will be designed using a mediator design pattern because it acts as a go-between for multiple delivery app interfaces and the fulfillment partner. Store representatives (employees and managers) will interact with a front-end Flutter app which receives orders from third-party delivery apps and these interactions will be permanently stored on a remote Amazon server (Fig. 1).

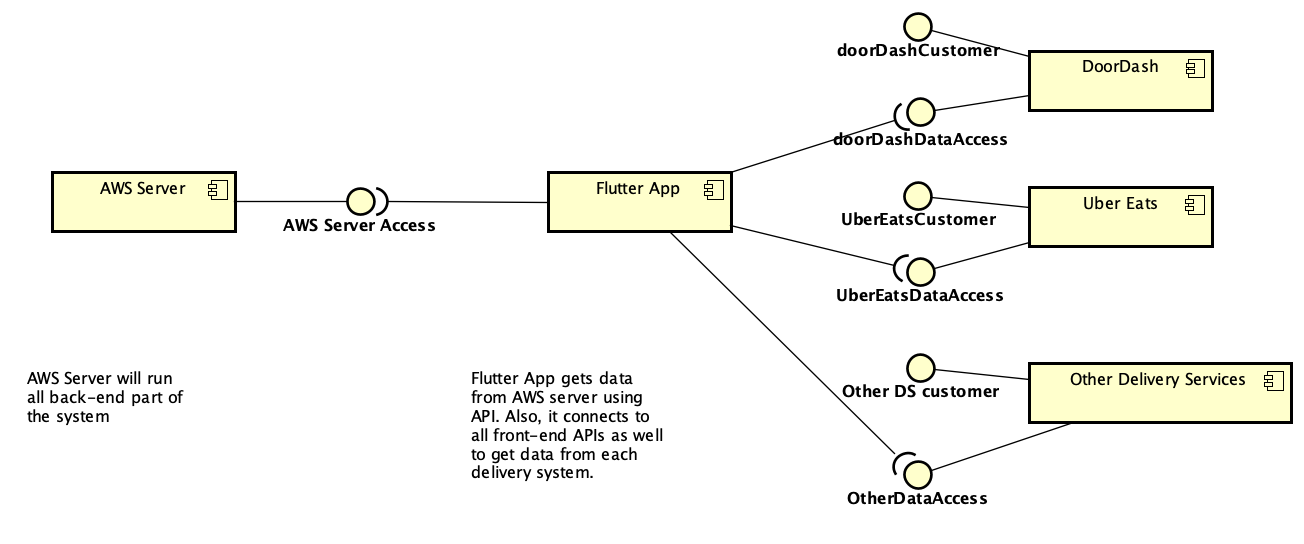


Fig. 1. Component-level conceptual design of the Speedy Snacks app.

We chose the mediator design pattern because it will facilitate communication between the different platforms and enable developers to tailor requests from each platform:

“Mediator is a behavioral design pattern that lets you reduce chaotic dependencies between objects. The pattern restricts direct communications between the objects and forces them to collaborate only via a mediator object.” [1]

The design envisions two levels of user access, employees and managers, and expected, common interactions with outside delivery app APIs (Fig. 2).

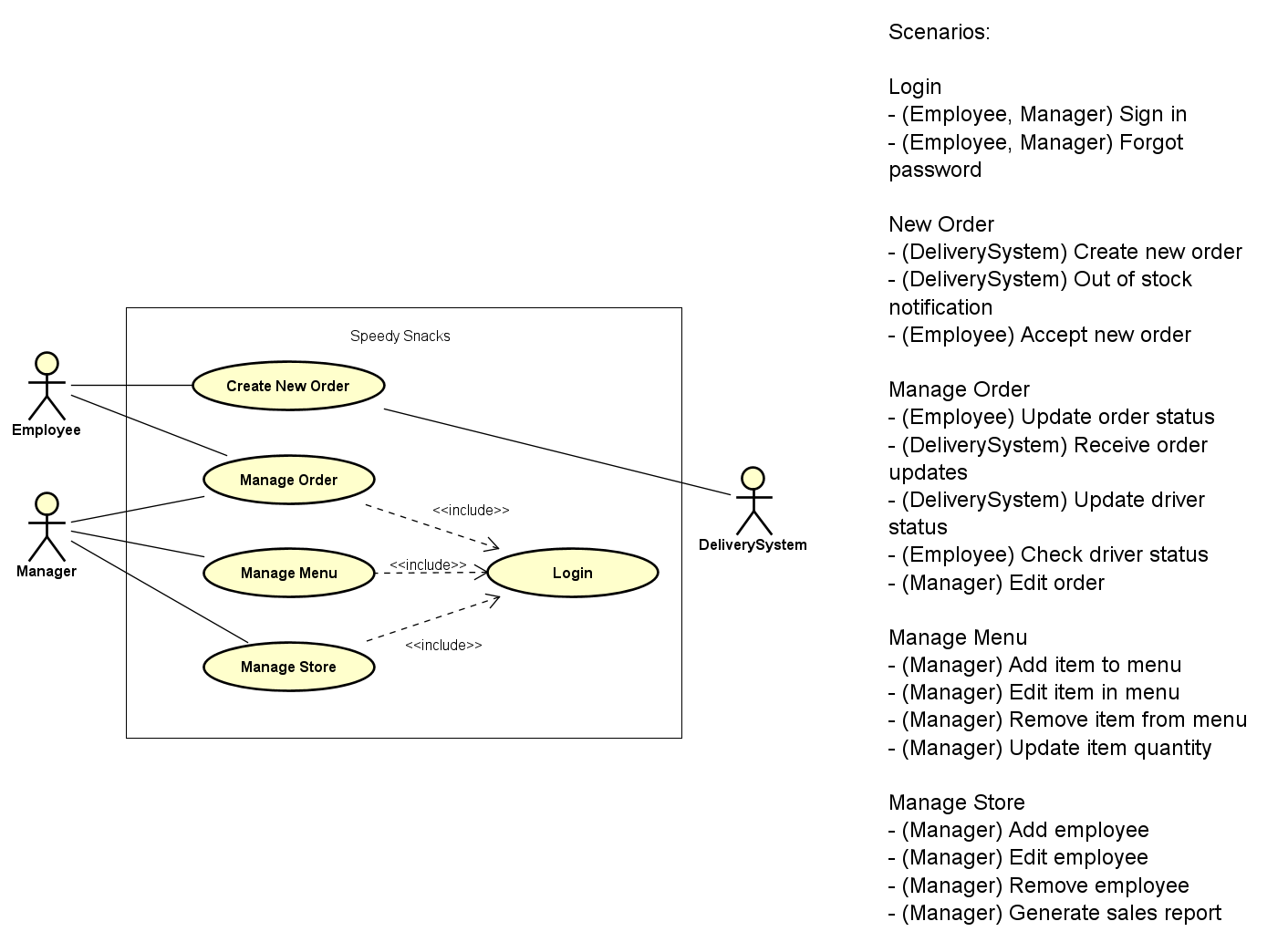


Fig. 2. Use case diagram (UC-1).

The login process will use a singleton design pattern to ensure that only a single user object can be created and implementations of “Manage X” processes will follow a model-view-controller design pattern. Although we aren’t currently sure which creational design pattern we’ll be using to turn incoming JSON objects into new order objects, it’s very likely we will use a factory, builder, or similar. We will also employ an observer design pattern to listen for new orders from the delivery app APIs.

TABLE III

USE CASE UC-1 DIAGRAM DESCRIPTION

| **Use Case** | **Actor** | **Description** |
| --- | --- | --- |
| Manage Order | Employee | Accept, edit, and cancel orders. |
| Manage Product | Employee | Update product quantity or details. |
| Manage Store | Employee, Manager | Add, edit, and remove products from the store’s menu. |
| Send Order | Employee, Delivery System | Send orders to the store. |
| Receive Order | Employee, Delivery System | Order receive to delivery system |
| Login | Employee, Store Management | Log in to the Speedy Snacks application. |

In object-oriented programming it’s important to identify the highest-level “nouns” in the description of the problem that the system is intended to solve. In this problem, a **purchase order** is received from a **delivery app** that contains one or more **products** off the **menu** from a nearby **store** which is then fulfilled by an **employee** or **manager** of that store (Fig. 3).

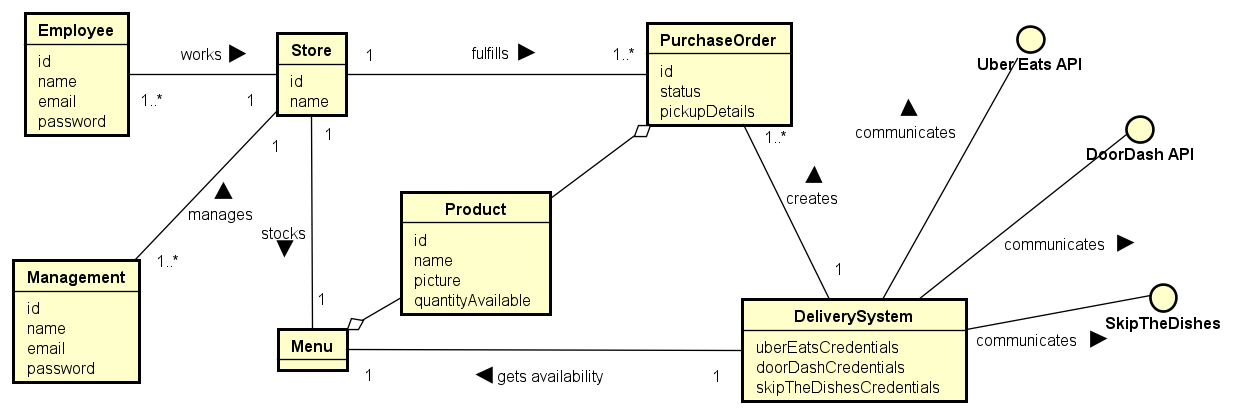


Fig. 3. Domain class model

This system will not be concerned with customer information (beyond what is necessary to verify the driver is picking up the correct delivery, which usually includes a customer name), payment transactions, and delivery driver logistics; these will be handled entirely by external systems.

## Alternate Design Possibilities (EM@FSE-b)

We created another use case diagram that we decided not to use because it was too detailed to be considered a component-level design (Fig. 4). There is also some confusion about the notation in some of the connections, such as Order Status.

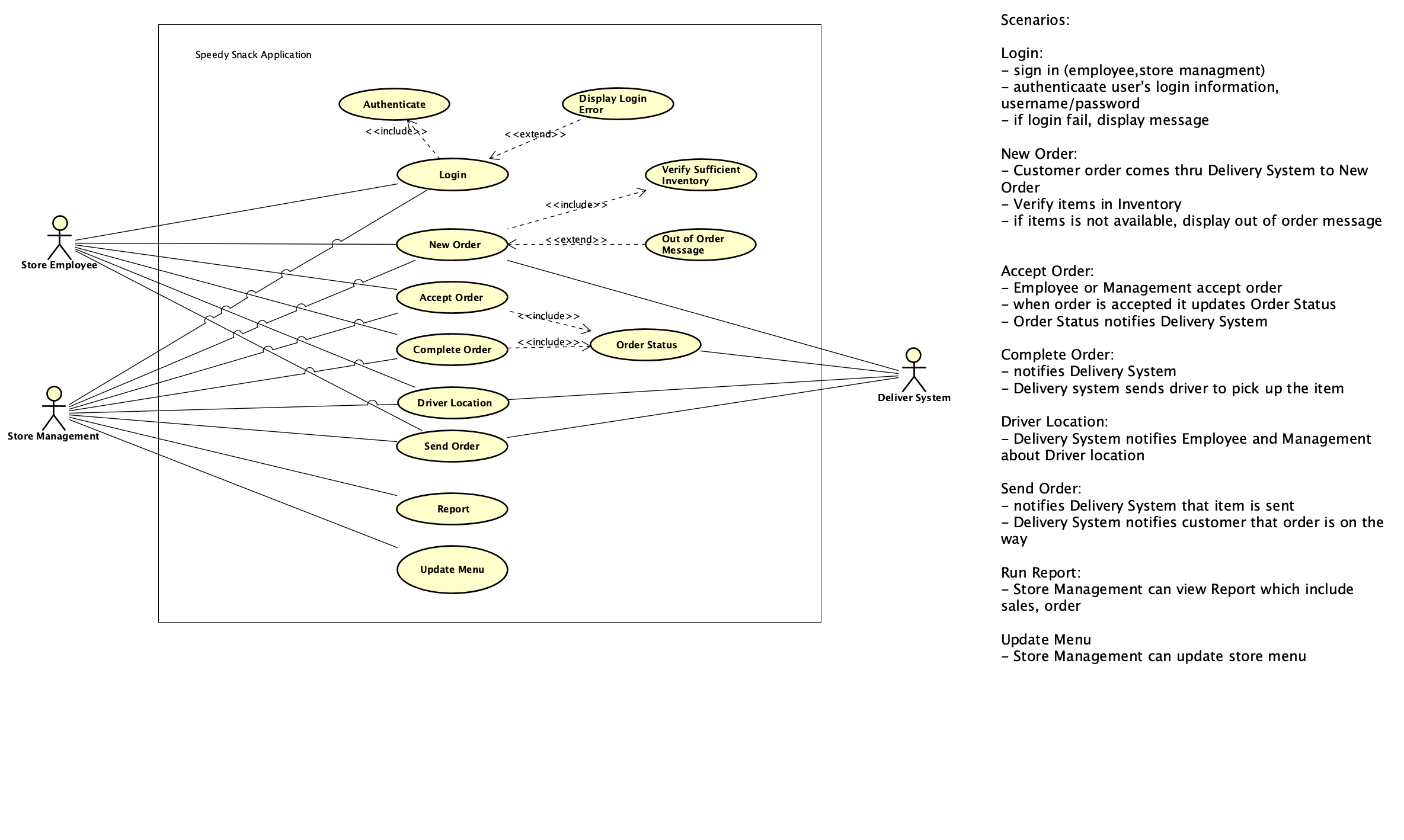


Fig. 4. Alternative use case diagram (UC-2).

The level of detail in UC-2 is premature in an Agile process, where customer requirements are expected to evolve over time. This fully developed design would be more appropriate within a truly waterfall framework.

TABLE IV

ALTERNATIVE USE CASE-UC2 DIAGRAM DESCRIPTION

| **Use Case** | **Actor** | **Description** |
| --- | --- | --- |
| Login | Employee, Manager | Logs into the Speedy Snacks application. |
| Authenticate | Employee, Manager | Checks login credentials. |
| Display Login Error | Employee, Manager | Displays error message when user enters the wrong username and/or password. |
| New Order | Employee, Manager, Delivery System | Receives new order from a delivery system and notifies store representatives it is available to view. |
| Accept Order | Employee, Manager | Store representative indicates they will fulfill a new order. |
| Complete Order | Employee, Manager, Delivery System | Store representative indicates the order is ready for the delivery driver to pick up. |
| Send Order | Employee, Manager, Delivery System | Delivery app API indicates the order has been picked up and is en route. |
| Driver Location | Employee, Manager, Delivery System | Updates employee or manager on the driver’s location upon request. |
| Report | Employee, Manager | Generates a new report from sales data. |
| Order Status | Employee,  Manager, Delivery System | Provides the order’s current status to the delivery app’s API or store representative. |
| Out of Stock Message | Employee, Manager, Delivery System | Displays a message when an item is out of stock. |

We chose to design the app with the use case UC-1 diagram based on our decision matrix (Table V). UC-1 and UC-2 have equal scores on Reliability, Maintainability, and Testability but UC-1 has higher scores on Portability and Reusability. The Portability score is higher for UC-1 because UC-1’s abstraction level is higher than UC-2, which will enable the developers to modify the source code throughout the life of the product. Similarly, UC-1 has a higher score on Reusability because greater abstraction allows developers to reuse code and incorporate polymorphism in the application.

TABLE V

DECISION MATRIX

| **Factors** | Reliability | Maintainability | Testability | Portability | Reusability | **Total** |
| --- | --- | --- | --- | --- | --- | --- |
| *Weights* | *5* | *4* | *1* | *1* | *3* |  |
| **UC-1** | 3 | 3 | 3 | 4 | 3 | **41** |
| **UC-2** | 3 | 3 | 4 | 2 | 1 | **37** |

# Implementation Strategy

## High-level Work Breakdown Structure (SER-1)

TABLE VI

WORK BREAKDOWN STRUCTURE

| **Task** | **Work Scope** | **Required Skill Set** | **Expected Time** |
| --- | --- | --- | --- |
| Project Plan | Initial project plan document with all sections completed | UML, design patterns, design processes | 1 week |
| Set up environment | Installation of MySQL, Python, Flutter, Android Studio | Git, VS Code | 1 1/2 weeks |
| Learn front-end languages and tools | Flutter crash courses | Familiarity with environment set up in previous task | 3 weeks |
| Design wireframes | Full “site map” defined and wireframes for all pages | Figma, design patterns | 2 weeks |
| Prototype GUI | Develop initial interface from all wireframes | Flutter | 4 weeks |
| Design business logic and persistent storage | Determine handling and return of API calls from in-house or third-party APIs, define necessary SQL tables | Flutter, Python, MySQL, database design, design patterns | 3 weeks |
| Learn back-end languages and tools | Research, test and increase proficiency in development of back end systems | Python, MySQL,  AWS, Django | 3 weeks |
| Develop front-end services | Develop API call logic from the front end, inventory lookup & contact form | Flutter | 6 weeks |
| Develop back-end services | Develop API hosted on AWS using websockets built on Python’s Django framework and MySQL | Python, MySQL,  AWS, Flutter, Django | 6 weeks |
| QA Testing | Analyze requirements, plan blackbox testing & report defects | Critical thinking, iOS & Android proficiency | 3 weeks |

## Schedule / Timeline (SER-1):

TABLE VII

PROJECTED TIMELINE

| **Due Date** | **Milestones** | **Challenges** |
| --- | --- | --- |
| 9/24/2022 | Project Plan | Inexperience with technologies |
| 10/5/2022 | Environment setup | Operating system differences |
| 10/26/2022 | Learn front-end languages and tools | Distinguishing necessary information from unnecessary information |
| 11/9/2022 | Design wireframes | Lack of clarity about back-end architecture |
| 12/7/2022 | Prototyping GUI | Meeting requirements with UI/UX Design |
| 12/28/2022 | Develop front-end services | Winter Break |
| 1/18/2023 | Learn back-end languages and tools | Winter Break |
| 3/1/2023 | Design business logic and persistent storage | Understanding middleware and data security |
| 4/12/2023 | Develop back-end services | Determining Django libraries to use. Validating inputs from delivery partners |
| 5/3/2023 | Completed project | QA Testing defects |

## Required Hardware (SER-1, EM@FSE-o):

The Speedy Snacks application will be tested on both iOS and Android. Most of the testing will be performed in emulators, but testing must include installation on at least one Android tablet. This tablet will be obtained by the sponsor from one of the delivery app partners, and the minimum specifications will be determined by that delivery service. It will be used by the sponsor to perform acceptance testing.

Team members developing and testing on Mac OS and Windows OS will be expected to use their personal computers to meet the minimum specifications for this project. ​The software will require a stable internet connection through Ethernet (LAN) or a wireless adapter (Wi-Fi). The processor speed must be at minimum 1 GHz, preferably 2GHz or more. An internal or external hard drive must have a minimum 32 GB available, with 64 GB or more recommended. The computer must have at minimum 1 GB of memory (RAM), with 4 GB recommended.

## Third-party content (SER-1, EM@FSE-o):

The technology stack for this project is open-source and has no licensing requirements for the development team. The sponsor will provide necessary computation and database resources (AWS). The app will be developed to use 3rd-party APIs such as Uber Eats, DoorDash, and Skip the Dishes but it will not be live until after the end of the project, so that only student API keys will be necessary to learn API syntax. If the project scope changes and a commercial API key becomes necessary the sponsor must provide it per the terms in the project prospectus.

Menu data, including pictures, will be (or has been) scraped by the sponsor and will therefore be considered “in-house” for practical purposes.

## Quality (SER-2):

Developers will write tests for their own code and fill out a code review checklist as part of their final submission for a task. The team will design and write whitebox tests for non-GUI code with the goal of **35% total line coverage** and blackbox tests for non-GUI code intended to smoke test the **50% of methods** representing key functionality. Developers of GUI code will design manual tests such that approximately 35% line coverage is achieved, with the understanding that this goal cannot be measured. (It will be left to the developer’s judgment whether they made a reasonable effort to achieve 35% line coverage, on the honor system.) Smoke testing of functional and nonfunctional requirements will be performed when the team demos system changes to the sponsor during sprint reviews.

We cannot at this time determine coding style requirements in languages we have not yet learned. Therefore, coding style will be determined during the designated learning periods (refer to TABLE VII) and added to a quality policy document that will be maintained on GitHub. Similarly, we have not settled on a continuous integration solution because the one with which we’re familiar, TravisCI, is not free. Our leading candidate is JetBrains because it has a free solution for students, and when this question has been decided we will update the quality policy with our process for using this tool.

Team members will work on tasks in discrete Task branches corresponding to Taiga tasks. [Commit messages will be well-written](https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/) and the optional "WIP" modifier will be used to signify when the commit is non-functional. Once functional and ready to be tested, a pull request will be made to merge the Task branch into the appropriate User Story branch. Once all tasks of a User Story are complete, the User Story branch will be merged into the dev branch. At the end of the sprint, all User Stories present in the Sprint Backlog should be complete and the dev branch will be merged into the master branch, signifying the submission of the deliverable. All pull requests must be reviewed by at least one team member, who will pull the branch to their local machine, run it, perform the developer’s tests, and submit a code review checklist when approving it. It will not be necessary to fill out a checklist when requesting changes.

Reviewers will be assigned in first name alphabetical order:

* Batbold will review Hillary’s pull requests
* Hillary will review Na’s pull requests
* Na will review Nicholas’s pull requests
* Nicholas will review William’s pull requests
* William will review Batbold’s pull requests

Upon opening a pull request, the reviewer has 72 hours to contact the developer with a plan to test, review, and either approve the PR or request changes. It will only be necessary to inform the developer when to expect the review to be complete, the review itself must only be completed by the end of the sprint.

## References/Sources of Information (EM@FSE-q)

The sponsor’s intention for this project’s scope is to design from the API documentation for three delivery applications: [UberEats](https://developer.uber.com/docs/eats/introduction), [SkipTheDishes](https://developers.flyt.io/api-reference), and [DoorDash](https://developer.doordash.com/en-US/docs/drive/tutorials/get_started/). It is vital to acquire and understand each unique API’s documentation, so that the developers are able to [scrape APIs](https://www.techslang.com/scraping-api-what-is-it-and-how-does-it-work/) to retrieve essential data from the delivery applications and formulate methods to interpret it.  
  
Because this project is expected to interact with at least three different delivery applications, the development team will build a REST API in Python and use the REST architecture to send and receive web service data. This new method of creating and using REST APIs with Python has been comprehensively documented in references like [intetrate.io](https://www.integrate.io/blog/an-introduction-to-rest-api-with-python/) and [realpython.com](https://realpython.com/api-integration-in-python/).  
  
The development team will use [Django](https://www.djangoproject.com/) during back-end development, because this project will use the Django REST framework. The development team will be expected to become proficient in this web development framework to build the project application’s API. The sponsor suggested beginning with two videos, a [quick start guide](https://urldefense.com/v3/__https://www.youtube.com/watch?v=ZsJRXS_vrw0__;!!IKRxdwAv5BmarQ!e90cC_aR0gH9DrJrxYFRsgUdXKngmIUSLNX1zGqm4V3ymom7WIMXkUr2GPgH2jeKvJN-UfvOkIonI9DtPw$) and an [educational introduction](https://urldefense.com/v3/__https://www.youtube.com/watch?v=e1IyzVyrLSU__;!!IKRxdwAv5BmarQ!e90cC_aR0gH9DrJrxYFRsgUdXKngmIUSLNX1zGqm4V3ymom7WIMXkUr2GPgH2jeKvJN-UfvOkIrmCJGcsQ$). References to the Django Rest Framework can be read at [realpython.com](https://realpython.com/django-rest-framework-quick-start/) or [stackabuse.com](https://stackabuse.com/creating-a-rest-api-with-django-rest-framework/).

Connections to the APIs will be formed using Websockets. This is the one technology the sponsor has no experience with, so the development team will need to learn it well enough to overcome blockers without on-call mentoring. [Websockets in Python](https://websockets.readthedocs.io/en/stable/) has its own library and the sponsor recommended beginning with [a video primer on Django integration](https://www.youtube.com/watch?v=cw8-KFVXpTE).

[Flutter](https://flutter.dev/) is a software cross platform development tool that will speed up the development process. The team will use Flutter during the front-end development phase. Because Flutter is a new software development kit for all the developers, they will likely need to learn it using tutorial websites such as [flutter.dev](https://docs.flutter.dev/get-started/codelab) or [codelabs](https://codelabs.developers.google.com/codelabs/first-flutter-app-pt1#0) to get started.

Notably, we will not have to learn AWS because the sponsor will set it up to save time.

## Scalability (EM@FSE-J)

The product is expected to launch at a small scale with less than 10 stores and fewer than 10,000 transactions per month. However, the sponsor has repeatedly stated that the app has great market fit and store owners adopted the previous app with great enthusiasm. They would therefore like to expand to 400 stores within the first two years by hiring a full-time sales and marketing professional. Key resources to support this effort are developer time and server costs. Because historical snack orders average approximately $100 (per the sponsor in our first meeting’s minutes), potential profits are high. Using a hypothetical margin of 1%, i.e. $1 of revenue per order, and assuming 400 stores would yield, conservatively, about 400,000 orders per month, this app would bring in about $400,000 per month. This would easily pay for server costs and operational costs, and would support a sales professional and three full-time developers with good pay and benefits. These projections are conservative because they are based on historical trends with the current app, and don’t account for potential partnerships with new delivery apps (e.g. UberEats) or potential viral adoption in a region.

A potential issue is that some cities may not have a sufficient base of delivery drivers, which will diminish the customer’s experience. If this happens, there is nothing we can do about it and our sponsor says they will pull out of markets of that type.

## Other Special considerations (ABET-2)

Our sponsor has worked on a similar product before called InstaSnack. He developed that application for his friend and therefore has very clear technical requirements and understanding. That gives us a great advantage because he has offered to help us get through technical blockers whenever possible, and he has committed to be available on Slack most of the time to answer questions.

# Process

## Process Description and Justification (SER-1)

The team will employ Scrum/Agile with two-week sprints, borrowing some planning elements (such as this project plan) from the Waterfall framework. Agile is a good fit for the customer’s requirements, a food delivery management app, because it is designed for small-scale, low-risk applications made by small teams with an emphasis on coding and low project management overhead. It is also familiar to the development team and facilitates frequent, open communication with the sponsor. However, one way in which it is a bad fit is that it’s designed for experienced developers who can anticipate how the project may evolve in response to technical constraints and issues.

This is where Waterfall planning is a good fit, because the sponsor has very clear technical requirements and previous experience building a similar app, and can plan out the development stages for the team in advance. Therefore, this project will include Waterfall elements such as this project plan, which includes a timeline of milestones and a definite end date, and separate phases of development such as design, coding, and testing.

## Tools (SER-1, EM@FSE-O)

Team members will all develop in a standardized environment as much as possible. We have chosen the popular, lightweight [VS Code](https://code.visualstudio.com/) as our IDE. Running and testing the Flutter app in an Android emulator and an iOS emulator will require [Android Studio](https://developer.android.com/studio) and [Xcode](https://developer.apple.com/xcode/), respectively. [Figma](https://www.figma.com/) will be used to create wireframes and, where feasible, generate GUI code.

Team meetings will be conducted in [Zoom](https://zoom.us/). Documents will often be created using [Google Docs](https://docs.google.com/document/u/0/) using its synchronous editing feature, but they will be hosted in the team’s Dropbox (see subsection Location of Project Artifacts). Collaboration, hosting, and version control of code and other deliverables will be managed using [Git](https://git-scm.com/), an open source distributed version control system.

## Roles and Responsibilities (SER-1):

The roles associated with the project’s development process are Product Owner (the sponsor), Scrum Master (a student, and Git Master (a student). No role will be permanently assigned to a team member throughout the capstone duration, they will be reassigned during sprint planning meetings. Assignments will go in first name alphabetical order starting with Batbold as Scrum Master and Na as Git Master. After Batbold is Scrum Master Hillary will be the second Scrum Master, and so on. After Na is Git Master Nicholas will be the second Git Master, and so on. This gives all team members the opportunity to practice both roles. The responsibilities of the [Product Owner](https://www.scrum.org/resources/what-is-a-product-owner) and [Scrum Master](https://www.scrum.org/resources/what-is-a-scrum-master) are described in the Scrum Guide [2]. It is the responsibility of the Git Master to enforce the Git policies described in the Quality subsection above and to perform merges from the dev branch into the master branch.

## Location of Project Artifacts (SER-1):

The project code and documentation will be hosted on [the team’s GitHub repository](https://github.com/nklee4/SpeedySnacks_Capstone2022) for collaboration. GitHub is the industry standard and unlikely to experience outages, and the sponsor and all team members are familiar with it. Scrum processes will be managed from [the Speedy Snacks Taiga board](https://tree.taiga.io/project/nklee4-speedy-snacks/), an open-source Agile project management tool. It is intuitive to use and the development team is familiar with it from previous classwork. It experiences occasional outages but it was chosen because it is free. Documents created only for internal use by the development team, such as daily standup meeting records, will be stored on [the team’s Dropbox](https://www.dropbox.com/home/Speed_snacks_group_documentation). It was chosen for ease of use, familiarity, reliability, and because upgraded accounts are free to students.

## Sponsor communications (ABER-3)

Meetings between the development team and sponsor/product owner will be held every two weeks on Wednesdays from 5:30 PM to 6:30 PM EST through [Google Meet](https://meet.google.com/). Other communications will be through [the Speedy Snacks project channel](https://app.slack.com/client/T04272WG3EJ/C042S9XQRA5), and it’s expected that team members will respond on Slack within 24 hours of sponsor communication. ASU account emails must also be answered within 24 hours, but these should be avoided when possible because we’ve already had problems with important messages going to spam.

# Risk management

## Identified Potential Risks (SER-2)

Risks affecting completion within the requested scope and timeline are present in all aspects of the project. We identified risks in group dynamics, sponsor communication, reliance and access to infrastructure, design limitations, implementation blockers, and requirements. Potential risks are listed below in order of smallest to largest impact (Low, Medium, High, Very High) along with the expected incidence rate.

*Low Impact*: There is a risk of the **team being unable to find time to meet synchronously due to scheduling conflicts.** The expected incidence rate is low as the team and sponsor are in the same time zone and all have moderately flexible schedules. There is also a potential risk of **interruption in the use of systems and tools needed in development.** This is a risk in any development project and our toolchain will have an expected incidence rate around average for projects of similar scope (moderate) which can’t be mitigated. Some tools are local and some are external third-party services which creates multiple points of failure in the toolchain due to the reliance on internet connection, future tool upgrades and maintenance, as well as team members’ personal hardware failures. (For example, one team member has lost power at the time of writing due to a hurricane.)

*Medium Impact:* The expected incidence rate of **communication issues (information did not reach a team member or the sponsor, personality conflict, breaches of professionalism, etc)** is low. Slack has been designated as the main channel of communication for all team members so that communications will not be missed. Personality conflicts could potentially have a moderate incidence rate; however, team members are required to follow a set of professional standards established by ASU. The expected incidence rate of **absent team members due to scheduled vacation, illness, holidays, or quitting** is moderate. We are human and sometimes we get sick, need time off, or have other pressing responsibilities. Between five team members and a sponsor, small disruptions are likely and unavoidable. However, the possibility of a team member outright quitting is unlikely because, as seniors, we are heavily invested in our education and looking forward to graduation.

The expected incidence of the **sponsor being unavailable to provide needed support and communication** is low. The expectation of sponsor communication is set to just once every two weeks. This gives the sponsor ample time to coordinate their schedule. The sponsor also made clear that they have high availability and prefer asynchronous communication in order to stay in the loop of all developments. Sponsor support is integral to the success of this project as the expected incidence rate of **team members lacking technical skills or experience needed for the project** is high and cannot be mitigated. We are students with varying levels of technical skill and expertise. Most of us have never used the required tools for the project and are not experienced application developers so we all will be expected to learn new tools and processes which will consume a significant portion of the project timeline.

*High Impact*: There is a risk that **the requirements for the project are too intense for timeline and scope.** The expected incidence rate is moderate as this is our first time working together as a team and on a project of such a large scope in our academic careers. It will be difficult to predict the intensity of some requirements, especially at the beginning of the project, until we become more familiar with our workflow and efficiency as a team.

There is also a high-impact risk of **individual time management issues** of which the expected incidence rate is moderate. We are all students in our senior year with heavy workloads, jobs, and families. While we all have experience with balancing classwork, this is a much larger project with less rigid scheduling which could take some time to adjust to. There is also a high-impact risk regarding **failure to adequately utilize the V-model/Agile/Scrum process.** The expected incidence rate is low as we all have experience with the Agile/Scrum process from previous coursework (e.g. SER 316). We will have to educate the sponsor how to be the product owner, who is unfamiliar with the Agile framework, and the team will have to individually ensure consistency in applying the tenets.

*Very High Impact*: A very high-impact risk to project completion is **losing documentation or source code.** The expected incidence rate is low due to using version control systems and programs such as Git, GitHub, Google Docs, Dropbox, and Taiga to develop the project. Individual progress could be lost on a team member’s local machine, but the potential setback is low because a working version of the project will persist on GitHub.

Another very high-impact risk is **misunderstanding the project requirements or broad changes to requirements late in the schedule.** The expected incidence rate is low as requirements will be thoroughly analyzed and vetted by all team members as well as the sponsor to ensure accurate interpretation. Changes to the requirements down the road are not expected but would have a much higher impact on the completion timeline.

A last, potentially very high-impact risk would be **changes to the project timeline by the sponsor.** The expected incidence rate is very low because this is a university capstone project that is required to be two semesters long. The sponsor understands this and has worked with at least six previous student groups, so they have experience with setting realistic goals for a capstone project. Extending the timeline would be impossible and shortening the timeline would incur potentially massive loss of app functionality.

## Mitigation Strategies (SER-2):

For each of the risks identified, the team has proposed both active and passive approaches to mitigate their incidence and cost as well as approaches to risks that cannot be mitigated fully.

*Active Mitigation*: To mitigate the risk of the **team being unable to find time to meet synchronously due to scheduling conflicts**, the team will work as hard as possible to secure a time where all members are available for weekly synchronous meetings. If this is not possible, we will utilize asynchronous communication through Slack chat whenever absolutely necessary. All daily stand-up meetings will be conducted through asynchronous Slack team chat. This should incur no additional cost to the project timeline or features.

If **communication issues (piece of information did not reach a team member or sponsor/personality conflict/breaches of professionalism)** arise,ASU provides professionalism and etiquette guidelines to help us navigate team and sponsor communication. Any consistent conflicts, after being addressed with the member personally by the team, will be escalated to the course coordinator for further guidance. All communication will go through the team and sponsor slack channels and team members are expected to check these daily. Weekly team meetings and bi-weekly sponsor meetings are to be attended by all team members. Any member that cannot attend due to unforeseen circumstances is expected to review the meeting minutes and promptly request any needed clarification from the team.

**Team members lacking technical skills or experience needed for the project** will be expected to spend time becoming acquainted with the tools and processes required.The team has created a timeline to set up the project development environment and become familiar with all tools needed for the duration of the project. We have devoted 1.5 weeks to setting up the environment, 3 weeks to learning front-end tools, and another 3 weeks for back-end tools. This is a necessary cost to ensure all team members are capable of handling the project. Team members and the sponsor have provided documentation and crash course material for each of the tools required. Any issues with the tools should be communicated in the daily stand-up meeting in Slack so that other team members can assist.

We will not be able to fully mitigate the risk that **requirements for the project are too intense for timeline and scope**. The team will develop a detailed product backlog to understand to the best of our ability the size and scope of each user story needed to complete the design and implementation. Any concerns with intensity will be communicated with the sponsor to assess which features are most important to the project implementation and whether any functionality should be decreased to complete the project within the timeline. As noted, the sponsor has had experience with several previous capstone groups and we are hopeful that this will translate into a realistic project scope.

To mitigate the team’s potential **failure to adequately utilize the Agile/Scrum process**, we will review it as we teach it to our sponsor. The team’s Slack channel has an automated standup meeting prompt each day at 7 AM reminding members to communicate what they worked on the previous day, what they plan to work on during the day, and any blockers they are experiencing so that the team may help to resolve them. All team members are expected to attend weekly progress meetings and document consistent progress on user story tasks using Taiga and GitHub, incurring no extra cost to the project timeline or features.

The team will utilize the distributed version control system, GitHub, to ensure all code and documentation is frequently updated and saved in order to mitigate the risk of **losing documentation or source code**. We will have a quality policy emphasizing frequent commits and outlining procedures for pushing code to GitHub to further reduce risk of losing code. This mitigation will not incur any cost to the project timeline or features.

*Passive Mitigation:* To mitigate the risk of **interruption in the use of systems/tools needed in development,** we have chosen popular, industry-standard tools that have a low risk of interruption of service. We will work as individuals and as a team to ensure that periodic interruptions in our personal work environments will not create a large impact in delivering project value. For example, the team member who is experiencing a power outage at the time of writing had communicated to the team beforehand that this would be a likely side effect of the hurricane winds.

We will not be able to fully mitigate the risk of **misunderstanding the project requirements or changes made to the requirements down the road**. However,because we are utilizing a Waterfall-style planning process, the team will be focused on clarifying requirements up front with the sponsor to ensure accurate interpretation. The sponsor set up a Slack channel so that we can quickly ask questions about requirements and resolve them quickly.

**Changes to the project timeline by the sponsor** are also a risk we cannot mitigate fully. The sponsor requires a working app by next May that can scale up to 400 stores in the next 2 years. Changes to the project timeline would mean sacrificing product features. As a team we would have to mitigate this to the best of our ability by working to adjust the sponsor’s expectations regarding what value we could potentially deliver in a shorter time frame.

We will not be able to mitigate fully a situation in which the **sponsor is unavailable to provide needed support and communication.** The team obtained detailed requirements from the sponsor at the beginning of the project to reduce reliance on sponsor clarification throughout the timeline. If the sponsor is unreachable for an extended period of time we will notify the course coordinator and move forward with the project as we understand it.

Team members must communicate to the team as soon as possible when an **individual time management issue** arises so that the team can quickly redistribute and complete the team member’s sprint tasks. Consistent time management issues will be addressed by the team and eventually escalated to the course coordinator if they persist. The team will adjust meetings as well as possible to accommodate planned absences.

We may or may not be able to mitigate fully the risk of **absent team members due to scheduled vacation, illness, holidays, or quitting.** If a team member quits, the team will notify the course coordinator and sponsor immediately. The team and sponsor will hold a meeting to incorporate the course coordinator’s guidance and adjust the project’s timeline and scope. Members experiencing illness must alert the team so that the workload can be redistributed.

# 

# References

1. “Mediator,” *Refactoring.Guru*. [Online]. Available: https://refactoring.guru/design-patterns/mediator. [Accessed: 22-Sep-2022].
2. “What is Scrum?,” *Home | Scrum Guides*. [Online]. Available: https://scrumguides.org/. [Accessed: 24-Sep-2022].