Slip Snapper: SRS

Project owner: Mvuyisi Scheepers

Affiliation: EPI-USE Labs

Capstone Group

Anoobis Software

Team Members	*Student Number*
CK (Christian) Devraj	u20504552
TS (Tshego) Manthata	u17110310
AI (Andrey) Omeltchenko	u04534205
RY (Regan) Zhao	u20556455
G (Gabriel) Grobler - Team Lead	u20534541

Table of Contents

1 Introduction

1.1 Product Overview

The Slip Snapper is a cross-platform application that assists in managing an individuals' expenses. It will accomplish this by allowing a user to scan their receipts using optical character recognition (OCR) with a mobile device and create comprehensive expense reports.

1.2 Purpose and Product Functionality

Our vision is to create a cross platform application that will allow a user to take a picture of a receipt and have the information extracted automatically. The app should also be capable of generating reports for a period such as a day, a week or a month. As the information being handled is of a financial nature and therefor a private nature, the user should be able to securely login to their account. One of our objectives as a team is to create the application in such a manner that if they wish to manage their expenses, such as editing the stored data, they will be able to do so.

The primary use of the application will be for a user that wishes to keep more detailed information on their daily expenditure. They could do this by using the generated reports to see in detail what their expenditure looks like. The project will not interact with any financial transactions other than keeping track of the information that is on the slips that have been scanned. That is to say it will not directly interact with any financial applications.

2 Class Diagrams

2.1 Illustration of the Slip-Snapper Class Diagram

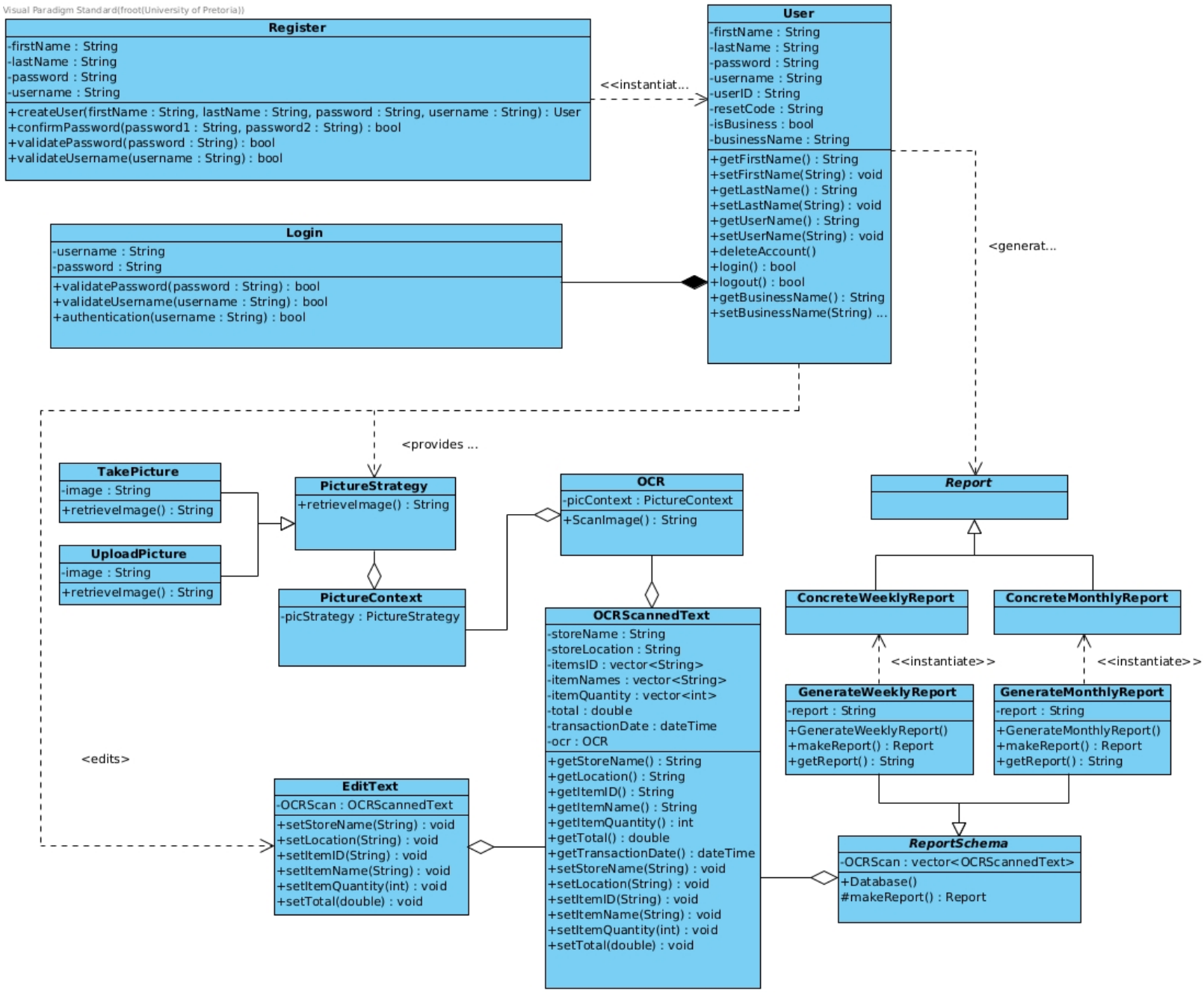


Figure 1: Slip-Snapper Class Diagram

2.2 Class Descriptions

- **User class**
  The User class maintains the User profile details. The UserID and Username attributes

hosted in this class will act as unique identifiers for the different users. The Login and Register classes both use this class to authorise and validate the User's entries namely the Username and password attributes. This class will also note if the user is a business or not. The list of attributes in this class include the "firstName", "lastName", "password" , "userID", "resetCode" and "userName" which are all of type String, the "isBusiness" variable is a Boolean.

- **Register class**
  The register class was created for the main purpose of validating the user credentials before a new account is created on the Slip-Snapper database. This is done by the class storing a String variable for the firstName,lastName,password and username. All these variables are passed into the createUser method which the returns a "User" object. However before this is done the system must perform the validateUsername and validatePassword methods and return "true" for successfull validation. "validateUsername" will ensure that the username is not already being used by another user and "validatePassword" ensures that the password is secure and safe.

- **Login class**
  The Login class is responsible for authentication of users and validating the user credentials before giving away access to an account. This is done by storing 2 private String variables namely username and password. These variables are passed into the "validateUsername" and "validatePassword" respectively, each function will analyze the user credentials by querying the database and will return a Boolean on a successful or failed login attempt.

- **PictureContext class**
  The PictureContext is the Context class of the Strategy design pattern, it is confiqured with a PictureStrategy object and it maintain a reference to the Strategy object. It will define an interface that lets PictureStrategy to access its data.

- **PictureStrategy class**
  The PictureStrategy is the Strategy class of the Strategy design pattern, it declares an interface that will discern which subclass' "retrieveImage" method will be called. The Context uses this interface to call the concrete strategies namely, TakePicture and UploadPicture.

- **TakePicture class**
  The TakePicture class is a Concrete Strategy of PictureStrategy class. This class stores the String "image" which will contain a link to the image taken by then user. The "retrieveImage" function allows external classes to access the "image" variable. The image will be retrieved by triggering the devices camera through the application.

- **UploadPicture class**
  The UploadPicture class is a Concrete Strategy of PictureStrategy class. This class stores the String "image" which will contain a link to the image taken by then user. The "retrieveImage" function allows external classes to access the "image" variable. The image will be retrieved by triggering the devices storage through the application. The user will then choose an image to upload.

- **OCR class**
  The OCR class is responsible for performing Optical Character Recognition on the image chosen by the user. This class stores a private reference to the user specified image as "picContext". This reference will be parsed into the "ScanImage" method which scans the image and returns the content in a String format.
- **OCRScannedText class**
- **EditText class**
  The EditText class' primary function is to edit/omit the OCR contents that were scanned by the user. This is very necessary as OCR technology is not 100% accurate. The EditText class stores a reference to the a OCRScannedText object which will be edited by the following functions.SetStoreName, setTotal, SetLocation, SetItemID, SetItemName and SetItemQuatity are all methods which take in a String parameter and make the necessary changes to the scanned contents. The changes are then applied to the original "OCRScannedText" object.
- **Report class**
  The Report class represents the "product" in the Factory method Design Pattern being implemented, this class contains methods and variables that will be accessible by the concreteClasses.
- **ConcreteWeeklyReport class**
  The ConcreWeeklyReport class is a concrete product in the Factory Method design pattern. This class will implement the interface for the Product, Report class.
- **ConcreteMonthlyReport class**
  The ConcreteMonthlyReport class is a concrete product in the Factory Method design pattern. This class will implement the interface for the Product, Report class.
- **GenerateWeeklyReport class**
  This GenerateWeeklyReport class is a concrete creator in the Factory Method design pattern, this class is mainly responsible for returning an instance of the concreteWeeklyReport class. This is done by using the "makeReport" method which will return a Report object.
- **GenerateMonthlyReport class**
  This GenerateMonthlyReport class is a concrete creator in the Factory Method design pattern, this class is mainly responsible for returning an instance of the GenerateMonthlyReport class. This is done by using the "makeReport" method which will return a Report object.
- **ReportSchema class**
  The ReportSchema class is the creator class in the Factory Method design pattern. This class will serve as an interface to for the concrete creators namely GenerateWeeklyReport and GenerateMonthlyReport which will instantiate the products.

3 User Characteristics

3.1 User Types

3.1.1 Individual Shoppers

This type of user would be a generic everyday shopper. This includes users who are interested in keeping track of their personal expenditure, the user would also want to identify potential areas where they could save money.

3.1.2 Small Businesses

These users would be interested in using Slip Snapper to keep track of the company expenses by logging their receipts. This can be very effective for basic accounting purposes as Slip Snapper will provide some statistics regarding the businesses expenditure.

3.2 User Stories

As a **User** I would want:

- Slip-Snapper to keep track of the purchases I made.
- Slip-Snapper to create basic reports of my expenditure.
- Slip-Snapper to save old expenditure reports to compare monthly/yearly.
- Slip-Snapper to securely save my data as most items scanned would be sensitive information.

As a **Business** I would want:

- Slip-Snapper to keep track of the business purchases made.
- Slip-Snapper to create basic reports of my expenditure on a daily basis.
- Slip-Snapper to save old expenditure reports to compare monthly/yearly.
- Slip-Snapper to securely save my data as most items scanned would be sensitive information.

User Type	**Purpose**	**Experience**	**Technical expertise**	**Education level**
Individual	Use Slip-Snapper to manage personal purchases made and to create and maintain basic financial reports of personal expenditure on a timely basis.	The Slip-Snapper application is an easy-to-use app that requires no prior experience is required for the Slip-Scanner application.	clear picture taking English literate	English literate Basic device navigation
Business	Use Slip-Snapper to manage the business purchases made and to create and maintain basic financial reports of business expenditure on a timely basis.	The Slip-Snapper application is an easy-to-use app that requires no prior experience is required for the Slip-Scanner application.	clear picture taking Scanning abilities English literate	English literate Basic device navigation

4 Functional Requirements

4.1 Use cases

4.1.1 Launching the application

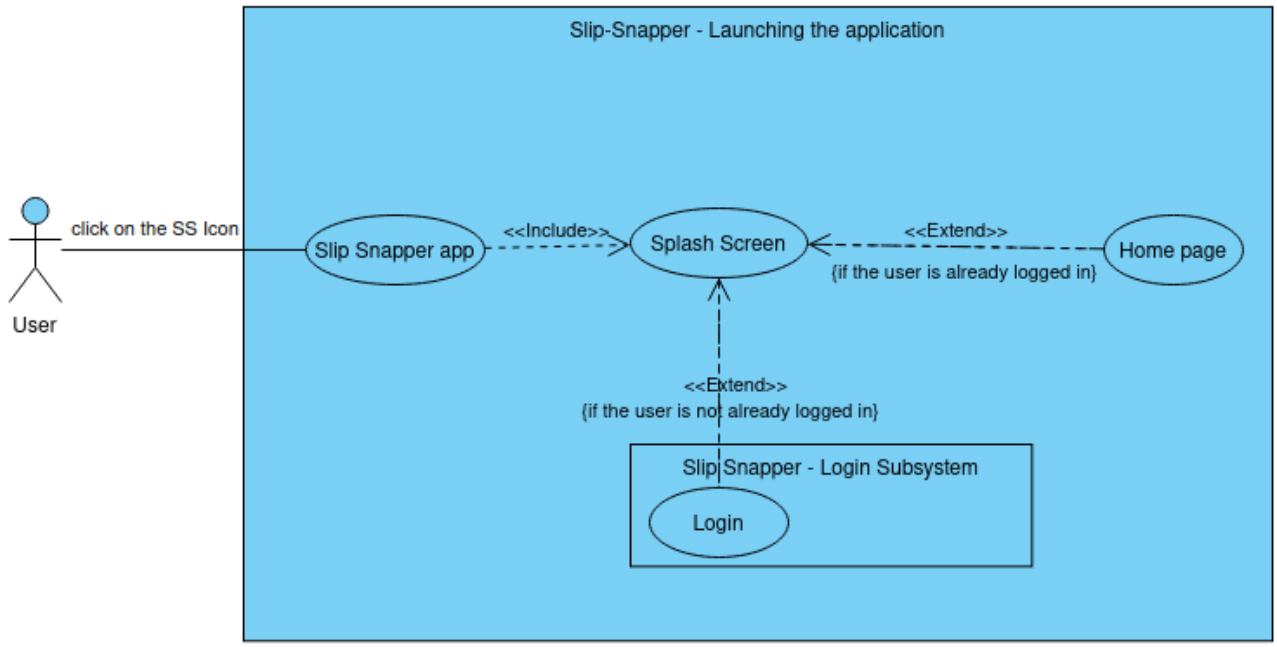


Figure 2: Use Case Diagram for Launching the App

- **Brief Description**
  This is how a user would typically launch the Slip Snapper application.
- **Basic Flow - Launching the App**
  1 The user will click on the Slip Snapper application icon.
  2 The user will first be greeted with the application's Splash page which will include the Slip Snapper and Epi-Use logos.
  3 The Slip-Snapper cross-platform app displays the Home page as the landing page if the user is already logged in and the use case ends.
- **Alternative Flows**
- **User is not logged in:**
  If the user is not logged into the Slip Snapper application, they will be directed to the Login screen*(Use Case 4.1.2)* when launching Slip Snapper.

4.1.2 Login

- **Brief Description**
  This use case describes how a user logs into the Slip Snapper application.
- **Basic Flow- login**
  1. The user inputs their credentials namely their username and password to log into the Slip-Snapper cross-platform app.
  2. The Slip-Snapper cross-platform app authenticates the username and password and logs the user into the system.
  3. The Slip-Snapper cross-platform app displays the Landing page and the use case ends.
- **Alternative Flows**
  - **Invalid Username/Password**
    If the user inputs their credentials incorrectly, the system will reject the login and output an error for the user.
  - **Forgot Password**
    If the user clicks on the forgot password button, they will be directed to a page where they can change their password once they are verified.
  - **Register**
    If the user inputs their credentials incorrectly, the system will reject the login and the user will be given the option to register a new account.*(Use Case 4.1.3)*
- **Preconditions** It is assumed that the user has already registered their account before attempting to login.

4.1.3 Register

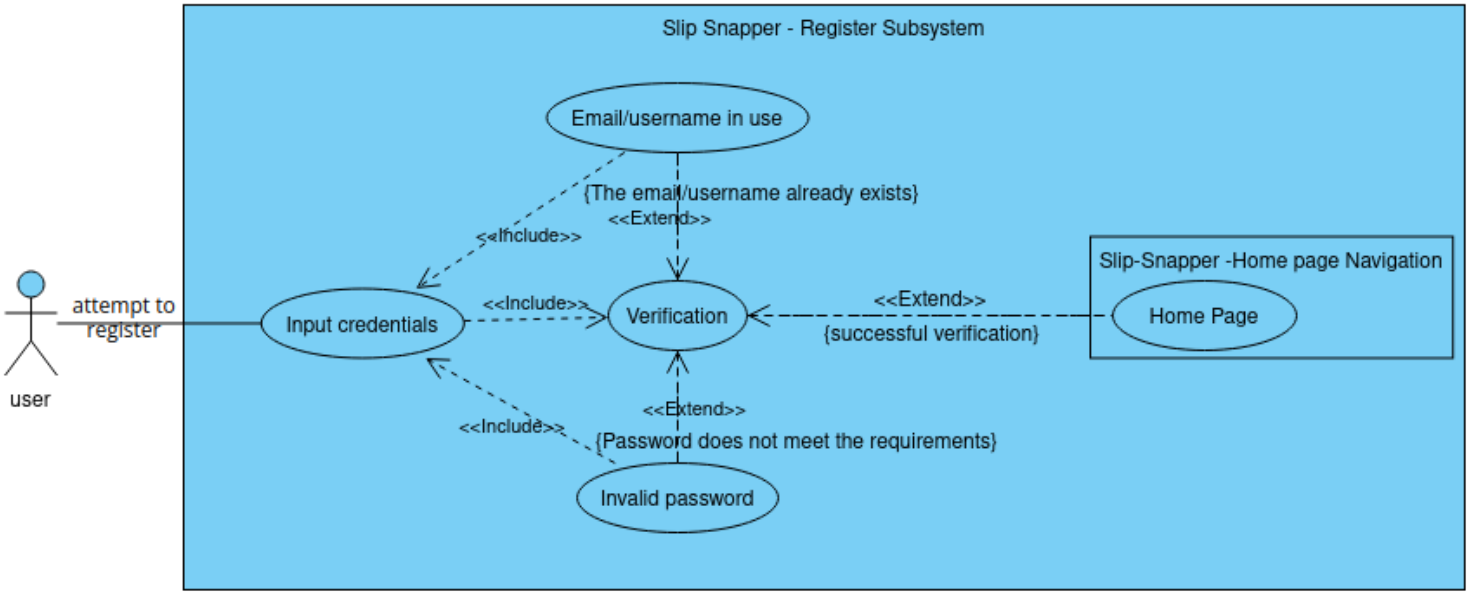


Figure 3: Use Case Diagram for User Registration

- **Brief Description**
  The register use case describes how a user would typically register an account on the Slip Snapper application.
- **Basic Flow- Register**
  1 The user inputs their credentials namely their username and password to sign up on the Slip-Snapper cross-platform app. The user will also be prompted to input other information such as their name and cellphone number.
  2 The Slip-Snapper cross-platform app verifies the username/e-mail and password.If verified the credentials will be stored on a database.
  3 The Slip-Snapper cross-platform app displays the Landing page and the use case ends.
- **Alternative Flows**
  - **Username already being used**
    The user tries to input a username that is already being used by another user, the system will output an appropriate error for the user.
  - **Invalid Password**
    The user tries to input a pass that does not meet the security requirements, this could be that the password is too short or is too simple for example. The system will output an appropriate error for the user.

4.1.4 Home page navigation

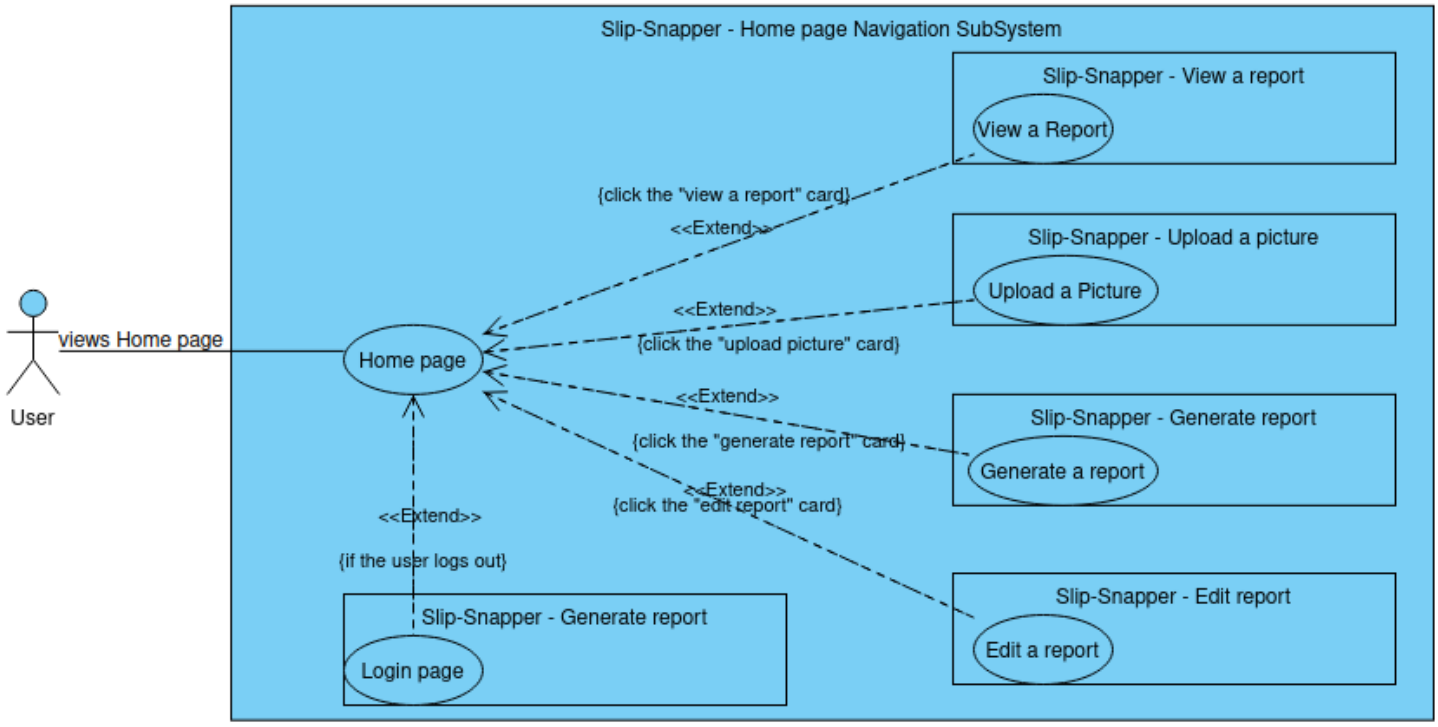


Figure 4: Use Case Diagram for Home page Navigation

- **Brief Description**
  This is how a user would typically navigate to the different features from the homepage on the Slip Snapper application.
- **Basic Flow - Home page navigation**
  1 The user will click on the "upload picture" button to navigate to the Upload Picture page.
  2 The user will click on the "generate the report" button to navigate to the Generate the Report page.
  3 The user will click on the "edit the report" button to navigate to the Edit the Report page.
  4 The user will click on the "view the report" button to navigate to the View the Report page.
- **Alternative Flows**
  – **User logs out**
  If the user is logs out then the Slip-Snapper cross-platform app will display the login page as the landing page.

4.1.5 Take a picture

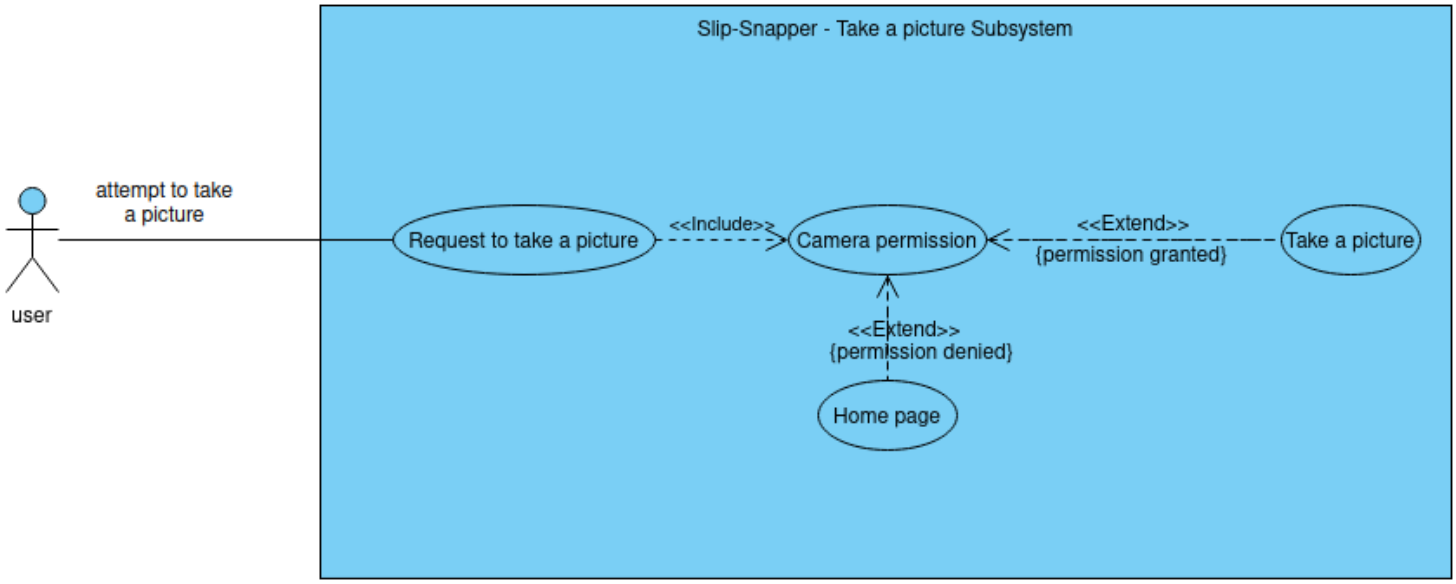


Figure 5: Use Case Diagram for Taking a Picture

- **Brief Description** This use case describes how a user would typically take a picture in-app using the Slip Snapper application.
- **Basic Flow**
  1. The user will attempt to take a picture of a slip in-app but be prompted to grant permissions to the device's camera.
  2. Upon granting permission the user will be able to take a picture of a slip.
  3. After granting permission the first time, the user can choose to "always allow" camera permissions to Slip Snapper.
  4. The user will then "take" the picture and OCR will be performed on the image.
- **Alternative Flows**
  The user denies camera permissions to Slip Snapper and the user will be directed back to the home page.
- **Post-conditions** If the user grants long term camera permissions to Slip Snapper, the app will no longer prompt the user to grant permission.

4.1.6 Upload a picture

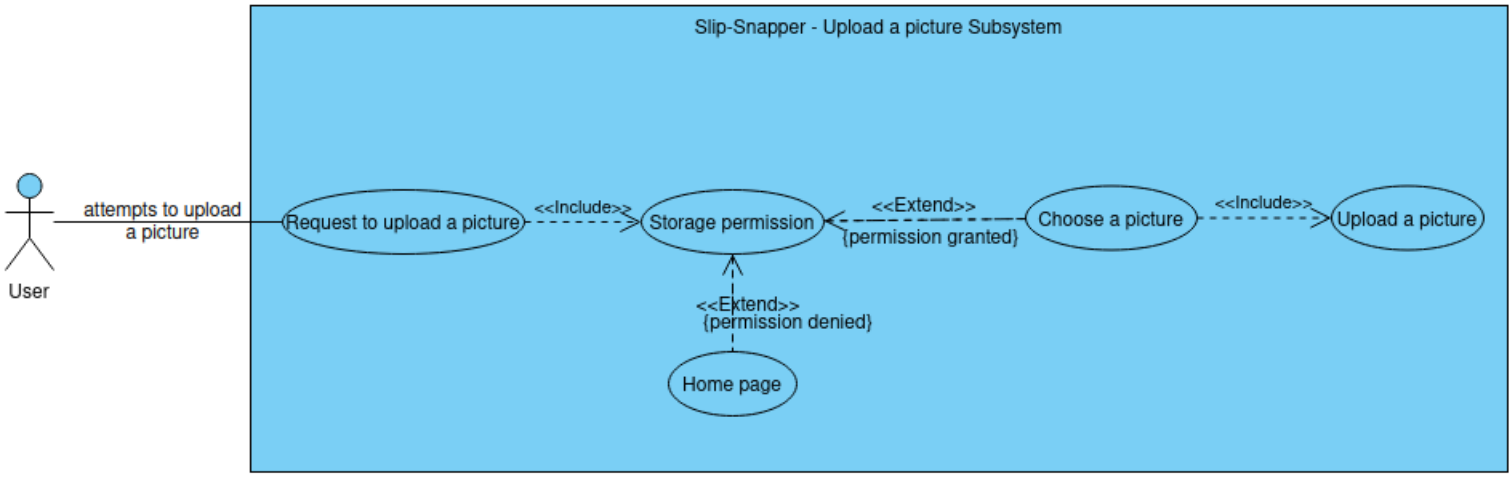
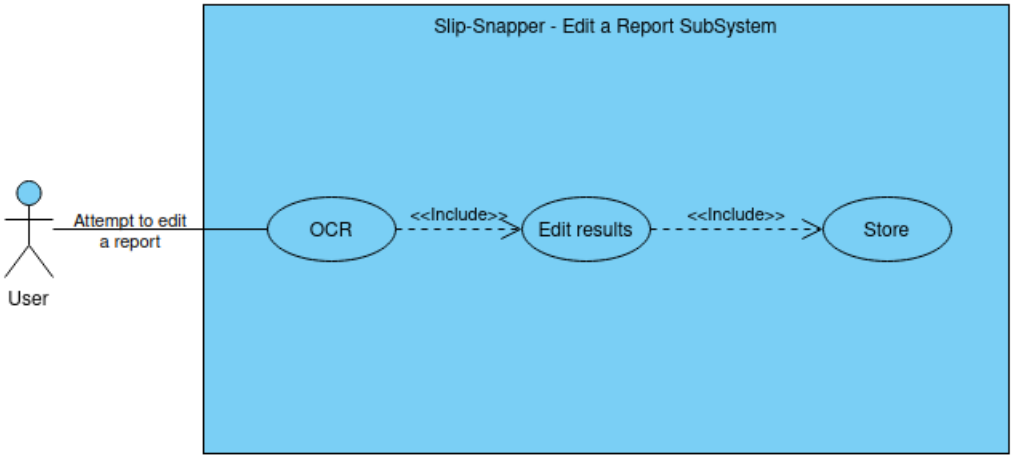


Figure 6: Use Case Diagram for Uploading a Picture

- **Brief Description** This use case describes how a user would typically upload a picture using the Slip Snapper application.
- **Basic Flow**
  1. The user will request to upload a picture from the device storage.
  2. The device storage permissions will be triggered to grant access to the app.
  3. The user will be able to select the desired picture from their device storage.
  4. The picture will then be uploaded.
- **Alternative Flows**
  The user may decide against uploading a certain picture, they can then cancel this task and be redirected back to the homepage.
- **Preconditions** It is assumed that the user has already selected an image before attempting to upload, however Slip Snapper will be ready to alert the user of this before upload.

4.1.7 Edit an item



.

Figure 7: Use Case Diagram for Editing a Report

- **Brief Description**
  Following an OCR scan on the respective slip, the user is given the opportunity to make changes to the scanned contents. This includes omitting/changing the names of products or the prices that were incorrectly scanned
- **Basic Flow**
  1 After the user has first scanned a slip using the OCR functionality.
  2 The user will be given the option to edit/omit the contents of the scanned output.
  3 The scanned contents is then stored on Slip Snapper.
- **Preconditions**
  The user first be required to scan a slip using OCR before they are allowed to edit the scanned contents.

4.1.8 Generate a Report

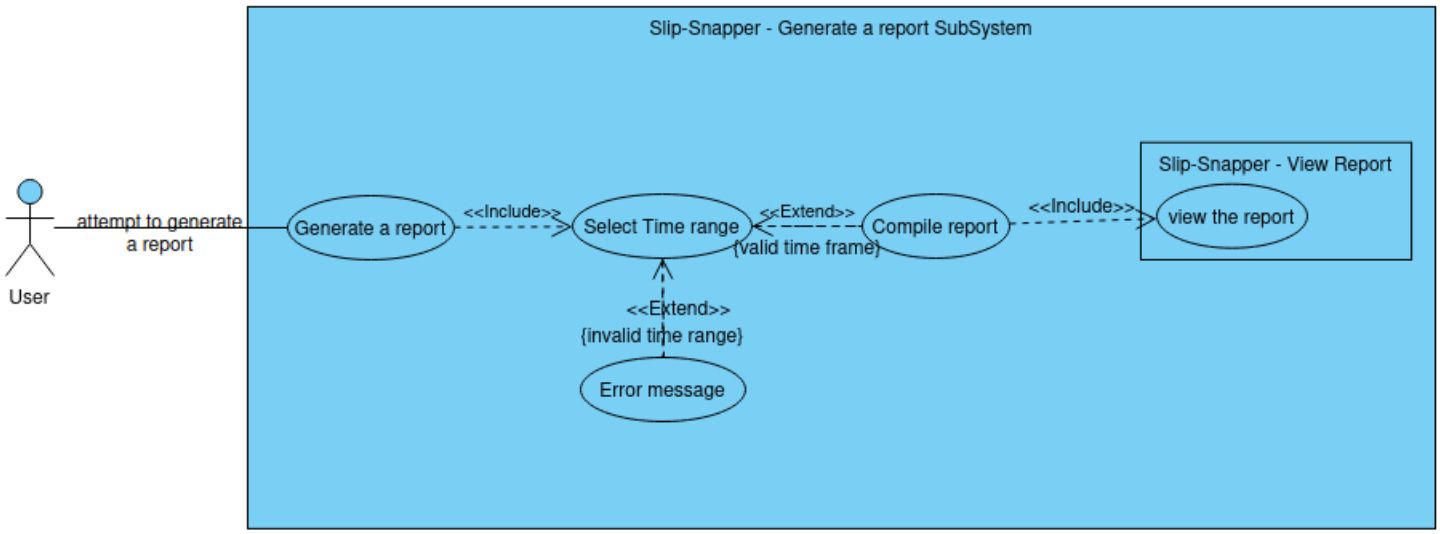
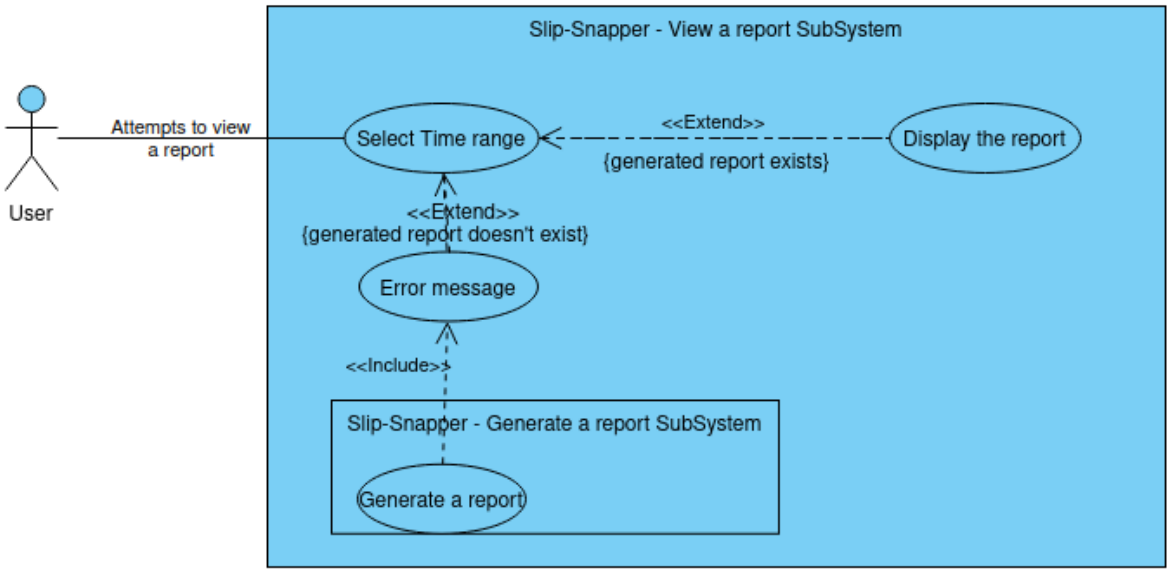


Figure 8: Use Case Diagram for Generating a report

- **Brief Description** This use case describes how a user would typically generate a basic expenditure report through the Slip Snapper application.
- **Basic Flow**
  1. The user will have the option to generate a report on the previously scanned slips.
  2. The user will be able to select the time period that the report must be generated. For example a report for May 2022 - June 2022.
  3. Slip snapper will then compile and display the report for the user.
- **Alternative Flows**
  The report may fail as the time period selected may not have any slips scanned. The user will then be prompted with an error message.
- **Preconditions**
  It is assumed that the user has scanned slips in the time period where the report will be generated, however Slip Snapper will alert the user if an error is encountered.

4.1.9 View a report



.

Figure 9: Use Case Diagram for Viewing a report

- **Brief Description** This use case describes how a user would typically view a basic expenditure report through the Slip Snapper application.
- **Basic Flow**
  1 The user will have the option to view a report that was previously generated.
  2 The user will be able to select the preferred report to view. For example, view a report for May 2022 - June 2022.
  3 Slip snapper will then display the report to the user.
- **Alternative Flows**
  If there are no generated reports, the user will then be prompted to generate a report first before viewing it with an error message.
- **Preconditions**
  It is assumed that the user has already generated the report before viewing it, however Slip Snapper will alert the user if an error is encountered.

4.2 Functional Requirements

FR1. The Slip-Snapper application shall ensure secure user access.

FR1.1 The Slip-Snapper application shall **ensure secure user registration.**

FR1.2 The Slip-Snapper application shall **ensure secure user login.**

FR1.3 The Slip-Snapper application shall allow the user to reset their password.

FR2. The Slip-Snapper application shall scan receipts (using AI, ML and/or OCR) from various businesses to determine various types of data:

FR2.1 The Slip-Snapper application shall derive the **store name or location** on a slip.

FR2.2 The Slip-Snapper application shall derive the **transaction date** on a slip.

FR2.3 The Slip-Snapper application shall derive the **item names** on a slip.

FR2.4 The Slip-Snapper application shall derive the **item quantities** on a slip.

FR2.5 The Slip-Snapper application shall derive **item prices** on a slip.

FR2.6 The Slip-Snapper application shall derive the **total price of the items** on a slip.

FR3 The Slip-Snapper application will allow the user to make the following changes to the scanned contents:

FR3.1 The Slip-Snapper application shall allow the user to **edit** specified items/prices on the slip.

FR3.2 The Slip-Snapper application shall allow the user to **omit** specified items/prices on the slip.

FR4 The Slip-Snapper application shall process and automatically categorize the scanned data before storing it.

FR5 The cross-platform app shall automatically generate daily, weekly, and monthly expenditure reports in PDF format.

FR6 The Slip-Snapper application shall allow for the management and correction of stored data.

FR7 The Slip-Snapper application shall allow the user to upload the generated reports.

5 Quality Requirements

- **[QR1] Scalability**

QR1.1 Slip-Snappers database will need to be scalable in future especially if businesses/-companies decide to utilize Slip-Snapper application. The database will need to potentially cater for hundreds of slips a day per business.

- **[QR2] Performance**

QR2.1 The Slip Snapper application will need to be very effective when performing its tasks. For example the OCR scanning will need to have a very high accuracy rate as it is cataloguing very important information.

- **[QR3] Availability**

QR3.1 Slip-Snappers database will need to be available +95% of the time. As Slip-Snapper will not be able to upload/save the slips scanned with OCR. Therefore this application will lose a great deal of its functionality is prove to be ineffective.

- **[QR4] Maintainability**

QR4.1 OCR Scanned Slips Database
Slip-Snapper must provide the functionality to edit/omit contents that are OCR scanned. This is mainly due to OCR technology not being 100% accurate and susceptible to errors.

QR4.2 Users Database
Slip-Snapper must provide the functionality to edit/remove users that are stored on the database. It is not uncommon for users to feel the need to change their username or remove their account altogether. Slip-Snapper must allow users to perform such tasks.

- **[QR5] Usability**

QR5.1 User Interface
The Slip-Snapper application must provide a friendly and enjoyable experience for users interacting with the software. The features provided by Slip-Snapper must be easily accessible by the user and it should meet all of the Usability Goals.

QR5.2 OCR Scanning
As a core functionality of Slip-Snapper is the OCR scanning of slips, this feature should be easily utilized and prompt the user to reuse Slip-Snapper for this functionality.

- **[QR6] Security**

QR6.1 User Authentication
If enabled by the user, Slip-Snapper must provide 2 step authentication to ensure it is not a hacker with malicious intent trying to access the user's slips.

QR6.2 Database Security
Since the data stored is of such sensitive nature, the database must provide almost impenetrable security. There must be no unauthorized access off the database through the application or externally.

6 Trace-ability matrix

Sub-systems 1					
Functional Requirements	Launching Application	Login	Register	Home Page Nav	Take Picture
FR1.1			X		
FR1.2		X			
FR1.3				X	
FR2.1					X
FR2.2					X
FR2.3					X
FR2.4					X
FR2.5					X
FR2.6					X
QR1.1	X				X
QR2.1	X	X	X		X
QR3.1	X	X	X		
QR4.1					
QR4.2		X	X		
QR5.1	X	X	X	X	X
QR5.2					X
QR6.1		X	X		X
QR6.2		X	X		X

Sub-Systems pt.2				
Functional Requirements	Upload Picture	Edit Item	Generate Report	View Report
FR3.1		X		
FR3.2		X		
FR4			X	
FR5			X	
FR6				X
FR7				X
QR1.1		X	X	
QR2.1		X	X	
QR3.1		X		X
QR4.1	X			X
QR4.2				
QR5.1	X	X	X	X
QR5.2				
QR6.1	X			
QR6.2	X			X

7 Service Contracts

7.1 Both Parties Contact Information

7.1.1 Anoobis Team

- Gabriel Grobler - u20534541@tuks.co.za
- Andrey Omolechenko - u04534205@tuks.co.za
- Christian Devraj - u20504552@tuks.co.za
- Regan Zhao - u20556455@tuks.co.za
- Tshegofatso Manthata - u17110310@tuks.co.za
- (TEAM MENTOR) Funmi Fagbola - funmi.fagbola@up.ac.za

7.1.2 Epi-Use Client

Johan Nel - johan.nel@epiuse.com

7.2 Outline of service and Team responsibilities

The Anoobis Software team has been tasked with designing,implementing and and producing the Slip-Snapper application. This project will need to be completed over 5-6 months. The Anoobis team will meet with the Epi-Use on a weekly basis to monitor progress and specify functionalities regularly.

7.3 Dispute Resolution and Remedies

7.3.1 Internal Anoobis Disputes

When settling most disputes or debates within the Anoobis team, is was decided that a vote would be carried out to arrive at a final decision. This will allow each member to voice their opinion and have some input in the decision making process. However in situations where a dispute is still not settled, if the concern relates to the project it will be brought forward to the client to make the final decision. If on the other hand the concern relates to a team members bad work ethic or unprofessional etc. The concern will be brought up to the team mentor, Ms Funmi Fagbola.