**BBUC\_Op1 – Begin Use Case**

Description: What should immediately happens after the user clicks ‘begin borrowing’ on the interface.

Pre-conditions: BorrowBookCTL class exists

memberDAO, loanDAO, bookDAO, display, cardReader, scanner, printer all exist

BorrowBookCTL added as listener to cardReader and scanner

BorrowBookCTL state == ***CREATED***

Post-conditions: BorrowBookUI is displayed

SwipeCard panel of BorrowBookUI displayed

Cancel button enabled

cardReader is enabled

scanner is disabled

BorrowBookCTL state == INITIALIZED

Data required: Requires data from user input on if they want to begin borrowing

**BBUC\_Op2.1 – Swipe Borrower Card (not restricted users)**

Description: Evaluates how the interface should respond once the borrower swipes their card. Information is gathered from the card swiped to identify the borrower which swipes their card and the user has no holds on their account.

Pre-conditions: cardReader visible and enabled

BorrowBookCTL added as listener to cardReader

memberDAO exists

BorrowBookCTL state == ***INITIALIZED***

Post-conditions: Scanning panel of BorrowBookUI displayed

Complete, Cancel buttons enabled

cardReader is disabled

scanner is enabled

borrower details displayed

existing loans displayed

scan count initialized as number of existing loans

existing fine message displayed if relevant

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: User Id gathered from users library card and if the user has any restrictions on their account

**BBUC\_Op2.2 – Swipe Borrower Card (restricted users)**

Description: Evaluates how the interface should respond once the borrower swipes their card. Information is gathered from the card swiped to identify the borrower which swipes their card. System then evaluates if the user has holds on their account

Pre-conditions: cardReader visible and enabled

BorrowBookCTL added as listener to cardReader

memberDAO exists

BorrowBookCTL state == ***INITIALIZED***

Post-conditions: Restricted panel of BorrowBookUI displayed

Cancel button enabled

cardReader is disabled

scanner is disabled

borrower details displayed

existing loans displayed

existing fine message displayed if relevant

overdue message displayed if relevant

atLoanLimit message displayed if relevant

borrowing restricted error message displayed

BorrowBookCTL state == ***BORROWING\_RESTRICTED***

Data required: User Id gathered from users library card and if the user has any restriction holds on their account

**BBUC\_Op3.1 – Scan Book (book not found)**

Description: Determines if the book can be borrowed. Since the book cannot be found, it cannot be borrowed by the user,

Pre-conditions: BorrowBookCTL class exists

scanner exists

BorrowBookCTL added as listener to scanner

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

Scanning panel of BorrowBookUI displayed

Cancel button enabled

cardReader is disabled

scanner is enabled

book not found error message displayed

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: Requires book ID, author and title of the book.

**BBUC\_Op3.2 – Scan Book (book not available)**

Description: Determines if a book can be borrowed in its state. Since the book is unavailable, the user cannot borrow it.

Pre-conditions: BorrowBookCTL class exists

scanner exists

BorrowBookCTL added as listener to scanner

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

Scanning panel of BorrowBookUI displayed

Cancel button enabled

cardReader is disabled

scanner is enabled

book not available error message displayed

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: Requires book ID, author and title of the book. Requires the books current state to see if it can be borrowed

**BBUC\_Op3.3 – Scan Book (book already scanned)**

Description: Scans the book and adds it to a pending loan list. However, since the book has already been scanned (either already on loan to someone else or the user re-scans it again), it will not be added to the loan list and the user will be given the error message.

Pre-conditions: BorrowBookCTL class exists

scanner exists

BorrowBookCTL added as listener to scanner

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

Scanning panel of BorrowBookUI displayed

Cancel button enabled

cardReader is disabled

scanner is enabled

book already scanned error message displayed

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: Requires book ID, author and title of the book. The data of the book is needed to see if the book is available to be scanned.

**BBUC\_Op3.4 – Scan Book (scan count < loan limit)**

Description: Scans the book and adds it to a pending loan list while the amount of books scanned is less than the loan limit

Pre-conditions: BorrowBookCTL class exists

scanner exists

BorrowBookCTL added as listener to scanner

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

Scanning panel of BorrowBookUI displayed

Cancel button enabled

cardReader is disabled

scanner is enabled

scanCount incremented by 1

scanned books details displayed

new (pending) loan created

(pending) loan added to pending loan list

Current pending loan list displayed

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: Requires book ID, author and title of the book. Data also needed is the amount of books already borrowed by the user to see if they can borrow more.

**BBUC\_Op3.5 – Scan Book (scan count == loan limit)**

Description: Scans the book and adds it to a pending loan list. Scanning stops once the user reaches the maximum amount of books they can have on loan at once.

Pre-conditions: BorrowBookCTL class exists

scanner exists

BorrowBookCTL added as listener to scanner

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

ConfirmingLoans panel of BorrowBookUI displayed

final pending loan list displayed

Cancel button enabled

Reject button enabled

cardReader is disabled

scanner is disabled

scanCount incremented by 1

BorrowBookCTL state == ***CONFIRMING\_LOANS***

Data required: Requires book ID, author and title of the book. Data of the books state is needed to know if it can be borrowed, and data for the amount of books the member has borrowed.

**BBUC\_Op4– Complete Scans**

Description: Finishes scanning the books being borrowed by the member

Pre-conditions: BorrowBookCTL class exists

Pending loan list exists

BorrowBookCTL state == ***SCANNING\_BOOKS***

Post-conditions: BorrowBookUI is displayed

Confirming Loan panel of BorrowBookUI is displayed

List of pending loans displayed

Cancel button enabled

cardReader is disabled

scanner is disabled

BorrowBookCTL state == ***CONFIRMING\_LOANS***

Data required: Data from the pending loans is needed for them to be confirmed

**BBUC\_Op5– Confirm Loans**

Description: Complete the borrowing book process

Pre-conditions: BorrowBookCTL class exists

Pending loan list exists

BorrowBookCTL state == ***CONFIRMING\_LOANS***

Post-conditions: Main Menu is displayed

All pending loans are committed and recorded

Loan slip of committed loans printed

cardReader is disabled

scanner is disabled

BorrowBookCTL state == ***COMPLETED***

Data required: The data of what books are being scanned and to whom are needed so they can be printed onto the printing slip. The state of the book and information on the members account is needed to determines if the loan can be confirmed.

**BBUC\_Op6– Reject Loans**

Description: Reject the current list of pending loans and start the scanning process again

Pre-conditions: BorrowBookCTL class exists

Pending loan list exists

BorrowBookCTL state == ***CONFIRMING\_LOANS***

Post-conditions: BorrowBookUI is displayed

Scanning panel of BorrowBookUI displayed

Borrower details displayed

Existing loan details displayed

pending loan list is empty

scan count == number of existing loans

Cancel button enabled

cardReader is disabled

scanner is enabled

BorrowBookCTL state == ***SCANNING\_BOOKS***

Data required: The results which determine if the user can borrow the books or not. For this use case to happen, the user needs to encounter something which causes their loans to be rejected, such as the book being unavailable or the user already having the maximum amount of books on loan.