

Student Loan System: Project Documentation

By Alexander Gale, Joshua Woods, Karen Farlow, Luis
Lemus



Kettering University
Class: CS-471, Software Engineering
Instructor: Jamal Alhiyafi
Term: B Section, Fall 2022

Product scope

Case Summary

We have been tasked by Dr. Jamal Alhiyafi (who will be referred to as the client) to develop a web application that will act as a banking portal where students, faculty, and affiliated banks will be able to access and manage financial data. Students will be able to request loans and review the ones they currently hold as well as manage their support checks. The banks need to review loans, approve or deny them, and access student financial information. The faculty will be able to directly bill the approved loans towards the student's tuition. A payment calculator for the students is also to be implemented for students to be able to review the long-term principle and interest of their loans.

This requires us to

Overview of system architecture

A client-server architecture will be used. The web host will have local access to the database for handling user and internal requests of data (queries). It will communicate through the internet using end-to-end encryption between the client and itself to satisfy the critical importance of security as a feature. This allows for fast response to system failures with a restart and lowers the general complexity of development. Client-server also lets us uphold the security using encryption in sending and receiving of data.

User Stories

From the case summary we isolated a multitude of user stories based on the necessity of the client. We can use these user stories to identify the requirements of the system, and will build it based on them.

The users will need to be able to log into the system

User Story: I, the user, need to be able to log into the system

Acceptance criteria:

Step	Expected Result
Access the webpage	Login screen is presented, with correct fields for logging in
Log in to account	When login information is entered, if it matches an account, the user is logged in and sent to the correct landing page. If it is incorrect it notifies them that it is an invalid login and offers a link back to the login page

Bank officers need to review loans for the students, this will require a secure connection as personal data will be handled

User Story: I, the bank officer, need to be able to review and approve student loans to fulfill loan applications

Acceptance criteria:

Step	Expected Result
Click the navigation link for the loan review page	The webpage for loan review loads up after the link is clicked, and the page successfully displays the pending loans
View the loan information for the pending loan requests	The loans displayed are pending and the information is correct
Accept or reject a loan	The loan is accepted or rejected correctly and that change is updated in the database

Banks want need access to approve or deny loan and they need a secure connection

User Story: I, the bank officer, need to be able to advertise to students to choose their bank

Acceptance criteria:

Step	Expected result
Click the navigation link for the advertisement submission page	The webpage for advertisement submission loads up after the link is clicked, and the page successfully displays the advertisement submission form
Enter advertisement information	The fields accept correct information
Submit advertisement	The advertisement is submitted to the database and is visible to system admins to review, with the correct submitted information

Student need access to able to apply for loans

User Story: I, the student, need to be able to apply for loans online

Acceptance Criteria:

Step	Expected result
Click the navigation link for the loan application page	The webpage for loan application loads up after the link is clicked, and the page successfully displays the loan application form
View options for the loan as part of the application form	The form has the correct fields, including those that specify which loan provider(bank) the loan is taken from
Submit a loan application	The application is submitted to the database and is visible for banks to review, with the correct submitted information

Student need access to able to apply for housing/support checks

User Story: I, the student, need to be able to request monthly checks for housing, so I have a place to live during the school year.

Acceptance Criteria:

Step	Expected result
Click the navigation link for the monthly housing check page	The webpage for monthly housing check loads up after the link is clicked, and the page successfully displays the housing check information
Enter the form information	The form has the correct fields, and accepts information correctly
Submit a loan application	The application is submitted to the database and has the correct information

Faculty need access to track student loans to be see who is keeping up with their loans and be able to advise them accordingly

User Story: I, as the college faculty, need to be able to track the progress of the student loans, so we can see who is keeping up on loan payments and who hasn't been.

Acceptance Criteria:

Step	Expected result
Click the navigation link for the student's loan information page	The webpage for student's loan information loads up after the link is clicked, and the page successfully displays the loan payment progress information

Faculty need be able to charge the loan directly to the student's tuition when a loan is approved

User Story: I, the college faculty, need to be able to request tuition payments from the banks, where students have approved loans.

Acceptance Criteria:

Step	Expected result
Click the navigation link for the tuition payment request page	The webpage for tuition payment requests loads up after the link is clicked, and the page successfully displays the tuition payment request form
Enter the form information	The form has the correct fields, and accepts information correctly
Submit the tuition payment request	The application is submitted to the database and has the correct information, and the request is visible by the banks.

Architectural analysis ('ilities' table)

Given that the web application handles sensitive data we felt it necessary to have a simple and robust system at the core, since the less components the better. Client-server is amongst the simplest that we found, followed by Three-tier architecture. Each of these models are designed to handle front ends and handle user requests, effectively narrowing down the possible models. We chose client-server because it was less complex than three tier, and our system did not benefit from the additional control of the third tier, meaning that client-server met our needs and was the best option.

Attribute	Importance factor	Importance	How Client-Server meets it	How Three-Tier meets it
Usability	3	<ul style="list-style-type: none">• The system needs to be fast and usable so that people can promptly get their loans when needed• System is intuitive to use and simple to navigate	<ul style="list-style-type: none">• Client-Server serves the intuitive client side, with few inbetween processes it can provide the responsiveness and speed needed	<ul style="list-style-type: none">• -Three-tier has an additional layer in comparison to client-server which will be slower, but the system is designed to handle customer interactions well which can be a good option.
Availability	2	<ul style="list-style-type: none">• -The system needs to be available at the beginning of the semesters so that students are not waiting on their loans.	<ul style="list-style-type: none">• The web server can run constantly, aside from scheduled downtime which can be planned for times when there is little to no traffic	<ul style="list-style-type: none">• -The web servers can run constantly, aside from scheduled downtime which can be planned for times when there is little to no traffic
Scalability	3	<ul style="list-style-type: none">• The scalability is of big importance so that as more banks get added and more students, the system does not	<ul style="list-style-type: none">• The expandability of client-server is primarily limited by the amount of concurrent users/data flow the server can handle at	<ul style="list-style-type: none">• Three-Tier can potentially handle higher bandwidth traffic, by splitting computation and information flow

		get under to much load and is able to handle everything	once and new hardware can be acquired	between layers
Performance	1	<ul style="list-style-type: none"> • The system does not need to be incredibly fast but needs to work as expected and provide the proper services 	<ul style="list-style-type: none"> • Client Server offers great performance since the data stores are in the same physical location as the web host 	<ul style="list-style-type: none"> • Three-Tier allows for higher backend performance through splitting the computation
Safety	1	<ul style="list-style-type: none"> • The system will not be a huge risk to the physical safety of the user 	<ul style="list-style-type: none"> • Client server relies on the server connection to operate, although the system losing operation is not a safety hazard. 	<ul style="list-style-type: none"> • Three-Tier relies on both server connection tiers to operate, although the system losing operation is not a safety hazard.
Security	3	<ul style="list-style-type: none"> • The system needs to be secure as it is very important that none of the users information or bank information get released to the public or are accessible by unauthorized users. 	<ul style="list-style-type: none"> • End-to-end encryption may be used to protect user data while traveling from client to server 	<ul style="list-style-type: none"> • The higher complexity makes data more vulnerable, but proper encryption protects the data equally to client-server
Maintainability	2	<ul style="list-style-type: none"> • The system needs to be maintainable if something were to go wrong it should be able to be fixed relatively easy 	<ul style="list-style-type: none"> • Maintenance of the server is all that is needed, updates to API can be used for the client automatically 	<ul style="list-style-type: none"> • Maintenance is also required of the extra layer

Reliability	3	<ul style="list-style-type: none"> • The system needs to be reliable, if someone is going for a 10k loan and accidentally get a 100k loan that would be a big problem 	<ul style="list-style-type: none"> • Reliability is limited only by the internet connection of the client and the reliability of the server, reliable web communication methods of verification of messages can confirm reliable data transmission 	<ul style="list-style-type: none"> • Reliability is limited by the third layer as well, although with modern data transfer standards the reliability of this communication layer are already very high
Portability	3	<ul style="list-style-type: none"> • The system will be portable as it is through a web application 	<ul style="list-style-type: none"> • Client server allows for each client to be developed individually, for each platform 	<ul style="list-style-type: none"> • Adding the third tier to the system would be able to leverage separation of client types more
Supportability	2	<ul style="list-style-type: none"> • The system should be easy to support but is not one of the most crucial 	<ul style="list-style-type: none"> • Support is available through the managed server 	<ul style="list-style-type: none"> • Support available through managed server and next layer

Fig. 1: An 'Ilities' table for the student loan system

Since the reliability, security and scalability were critical to our application and maintainability is also important, a server-client architecture will be adopted. This way ensuring the service is available will be easier to troubleshoot. In our case the web host is a local host, with access to the Kettering SQL server for its storage needs. This will be our main Server, which will handle user requests and responses. Since the data store we have is on the same network as the host, client-server architecture provides a more accurate picture of how the system will be laid out.

Architectural Pattern

The web host acts as the server, connecting to the client and sending/receiving information between the two of them. We ended up with a fairly standard and simple client-server setup as there was not any need for a more complicated system for the requirements to be met.

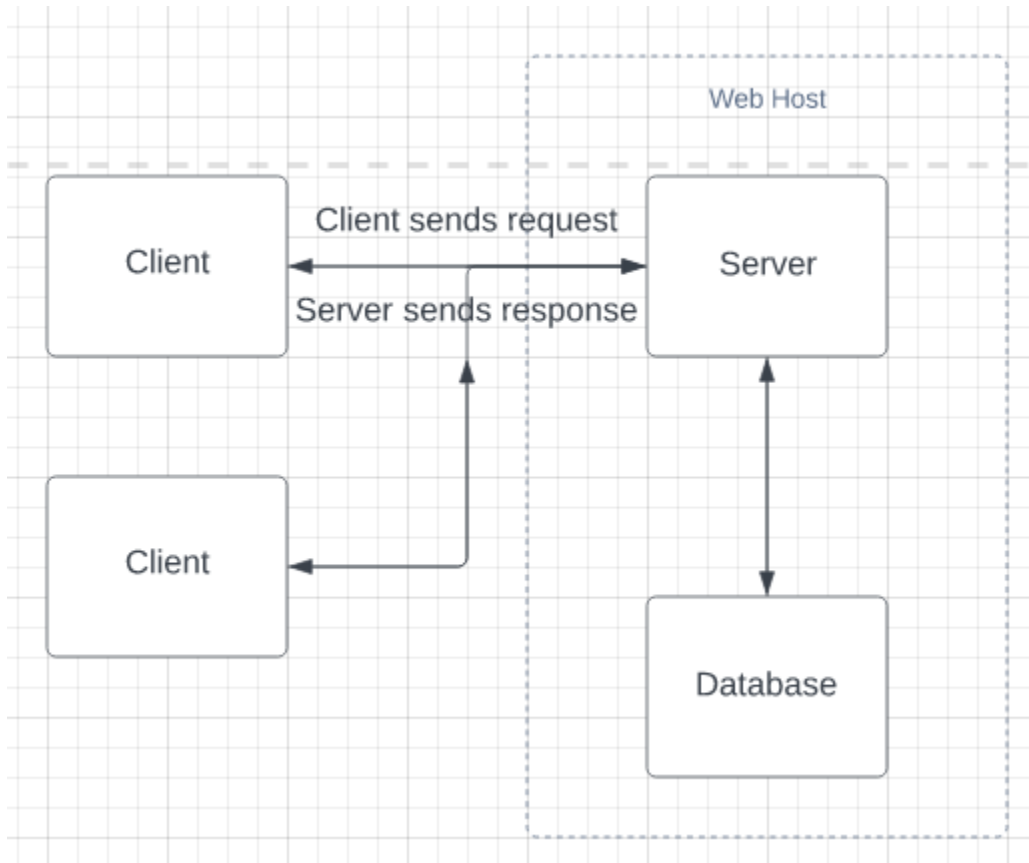


Fig. 2: An architectural pattern diagram for the student loan system

Class List (Stereotypes)

Here is a hierarchical view of the class list, separated by stereotypes. Student, Faculty, and Bank stereotypes are there to differentiate amongst the different user types and offer the proper functionality to each. Webpage boundary classes are implemented to separate each user into their appropriate user interfaces using their user type. The Login Page filters the user towards their correct Webpage. Each of the different webpage types (except for the login page) shows the users only the functionality that they have access to given their user type. A Form boundary class is used for the entity classes to interact with the Database information holder.

Entity Classes

The Student, Bank, and Faculty entity classes all inherit the information in the User class. This differentiates each user's capabilities. The Entity classes are all associated with the Form boundary class to submit or request information to and from the Database object.

The Loan class has a composition association with Bank and Student since the Loan cannot exist without its user or the Bank to provide it.

Boundary Classes:

Webpage boundary class will interface the Form with the Entity classes through a web-based user interface. The Form class handles user/server requests that require data to be queried from or added to the database with the support of the Database class.

Login, AccountCreation, Student, Bank, and Faculty boundary classes inherit from Webpage to obtain the session information of the user, they separate each user type's view

Information Holder Classes:

The Database class is the only class with the ability to interact with the database, it does so through Form requests. It will either provide or insert data into the database according to the Form's request.

Hierarchical view

Entity Classes:

- User
 - Student
 - Bank
 - Faculty
- Loan

Boundary Classes:

- Webpage
 - Login page
 - Account creation page
 - Student page
 - Bank page
 - Faculty page
- Form

Information Holder Classes:

- Database

CRC Cards

Using the above class list, we developed a set of CRC cards to use in explaining the interactions and roles between the parts of the system. Our CRC cards visualize the users on the system, what each user needs, and what information they can affect.

Class:	User
Responsibility:	Knows id Knows username Knows password Knows user type Can login Can logout
Collaborators:	Form, Login page

Class:	Student
Responsibility:	Can request Loan
Collaborators:	User, Form, Loan application, Loan advertising, Housing check request, Loan summary

Class:	Bank
Responsibility:	Knows bank ID Can set Loan status Can advertise Banks to Students
Collaborators:	User, Form, Bank page, Bank Landing, Advertisement application, Loan review, Housing check request approval, Tuition payment request approval

Class:	Faculty
Responsibility:	Knows faculty ID Knows Student database Knows Bank database Knows Loan database Can show Loan(s) for specific Students
Collaborators:	User, Form, Faculty page, Faculty Landing, Loan payment tracking page, Tuition payment request

Class:	Loan
Responsibility:	Knows loan taker Knows loan provider Knows loan date Knows loan amount
Collaborators:	Form

Class:	Webpage
Responsibility:	Knows current page
Collaborators:	User, Form, Student page, Faculty page, Bank page

Class:	Login page
Responsibility:	Knows login information Can authenticate Can redirect a user to their appropriate landing page
Collaborators:	Webpage, Student Landing Page, Faculty Landing Page, Bank Landing Page, Account creation page

Class:	Account creation page
Responsibility:	Knows values input into page Can create a user account
Collaborators:	User, Database, Form, Login page

Class:	Student page
Responsibility:	Knows Bank Advertisements Can redirect to a different we
Collaborators:	Student, Student landing page, Loan application, Loan advertising, Housing check request, Loan summary

Class:	Bank page
Responsibility:	Can redirect a Bank to Advertisement application page Can redirect a Bank to Loan review page Can redirect a Bank to Housing check request approval page Can redirect a Bank to Tuition payment request approval page
Collaborators:	Bank, Advertisement application, Loan review, Housing check request approval, Tuition payment request approval

Class:	Faculty page
Responsibility:	Can redirect a Faculty to Loan payment tracking page Can redirect a Faculty to Tuition payment request page
Collaborators:	Faculty, Faculty Landing, Loan payment tracking page, Tuition payment request

Class:	Form
Responsibility:	Can submit entries to the database Can query the database
Collaborators:	Database

Class:	Database
Responsibility:	Knows all long-term storage of system information Can accept entries from forms Can verify login information Can send requested information to web pages
Collaborators:	Form, Login page

Class Diagram

Using the class information from the list and the CRC cards, we then constructed a class diagram to show the flow of information through the system, and the relations between classes. The full diagram can be seen in Fig. 3 below

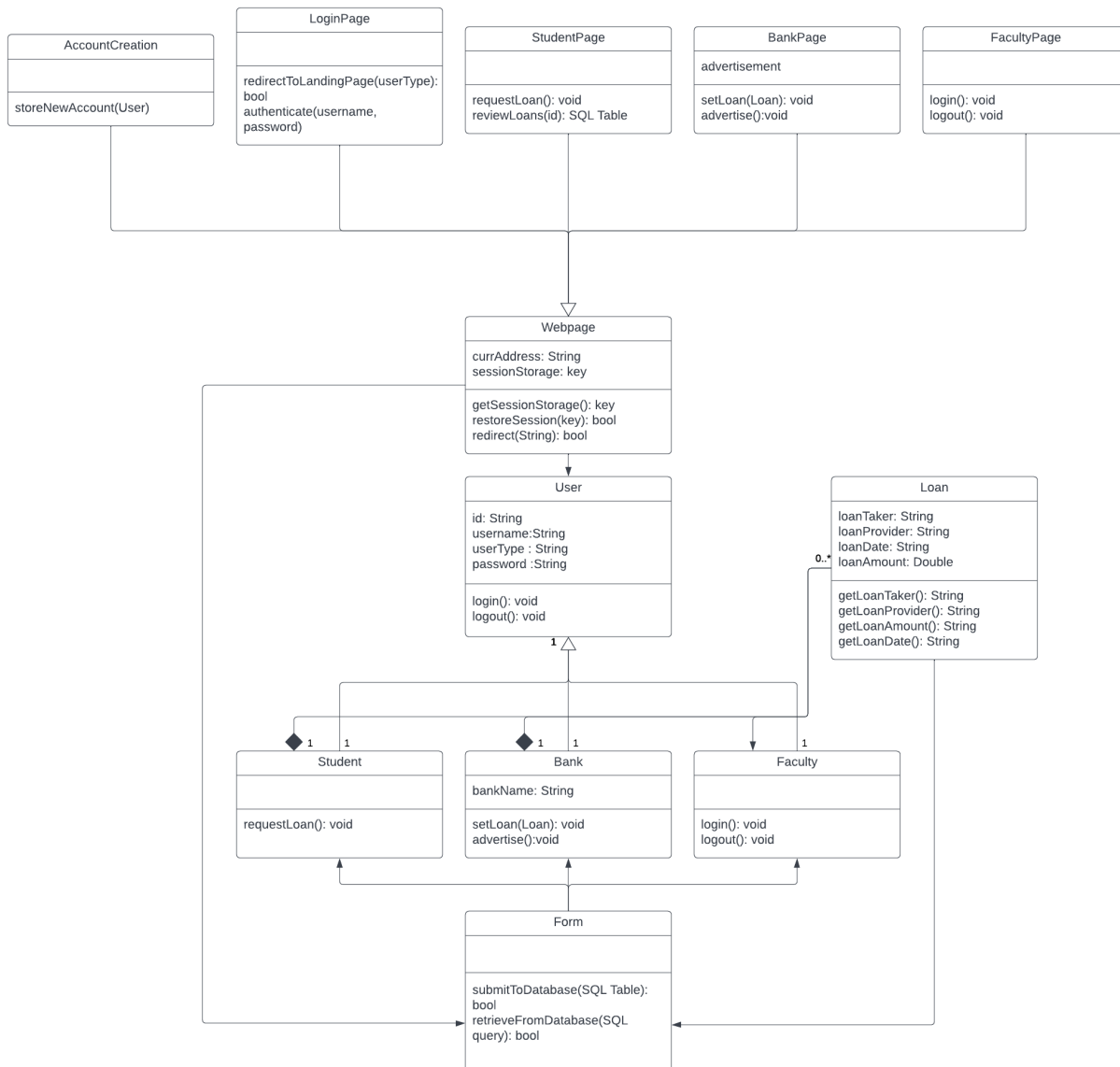


Fig. 3: The class diagram for the student loan system

Sequence Charts

In order to model the behavioral patterns of the system, we made use of sequence charts and a state diagram to model these patterns. Below are the Sequence diagrams for functionality that was completed within our Sprints so far which includes Web-Based Access, Loan Review, Loan Application and Tuition Payment Redirection systems

Web Based access for users (Fig. 4):

Here can be seen how the user is supposed to log in to the system from the Webpage class (specifically the Login page which extends it) by sending a message (encrypted) to the web server to validate the login information and return the login success or failure.

Web-based access for 3 types of users

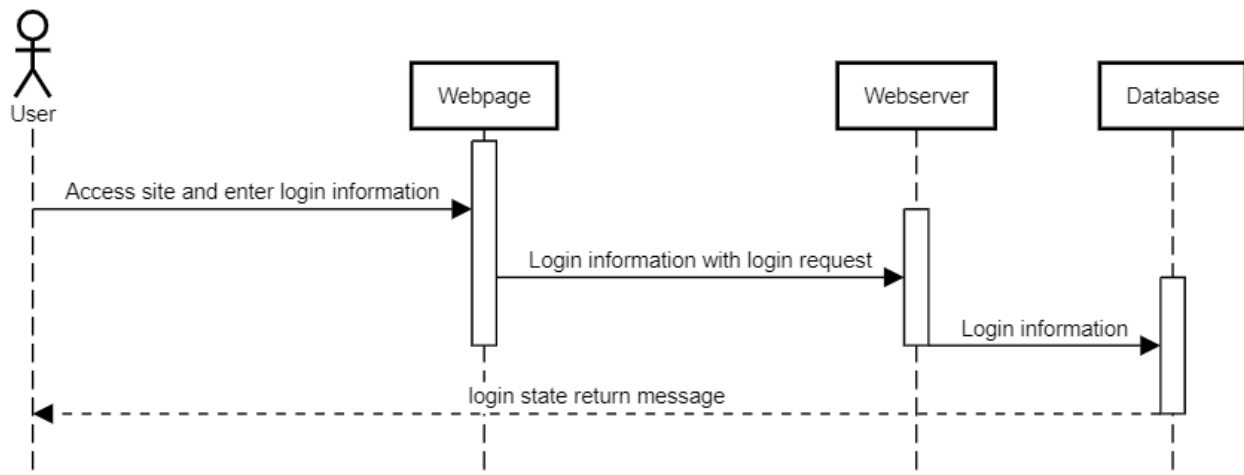


Fig. 4: The sequence diagram for web-based access

Loan review (Fig. 5):

The Bank can request to see the loans pending approval through the Webpage class (specifically the Bank page subclass) by sending a message to the web server to query the database and return any matching Loan information

screens for loan officers to access the system for reviewing loans

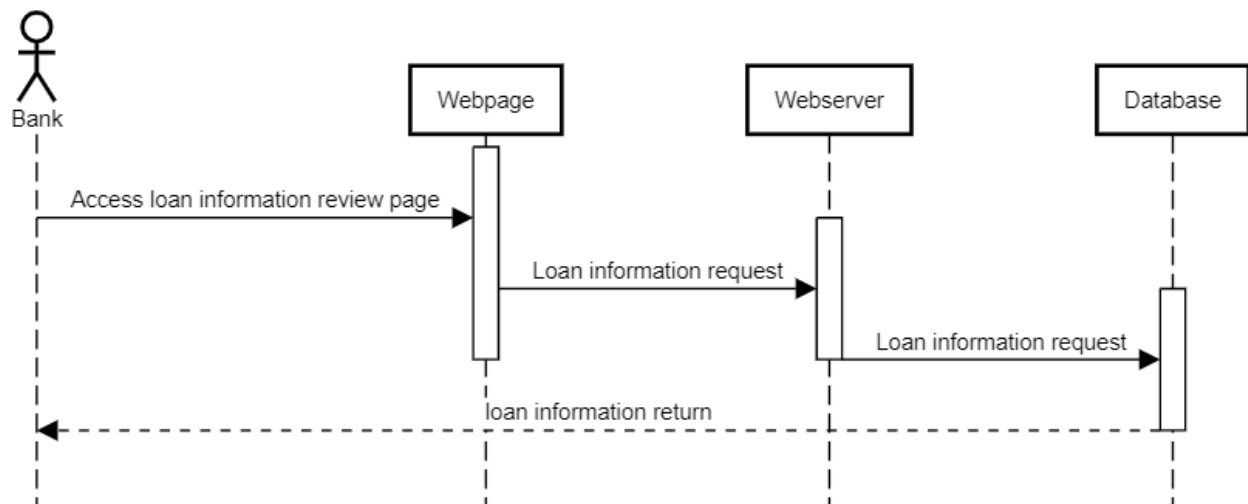


Fig. 5: The sequence diagram for loan review

Loan application (Fig. 6):

A Student can request a loan by sending a message through the Webpage (specifically the student page) to be submitted to the database by the web server.

screens for students to access the system for applying for loans

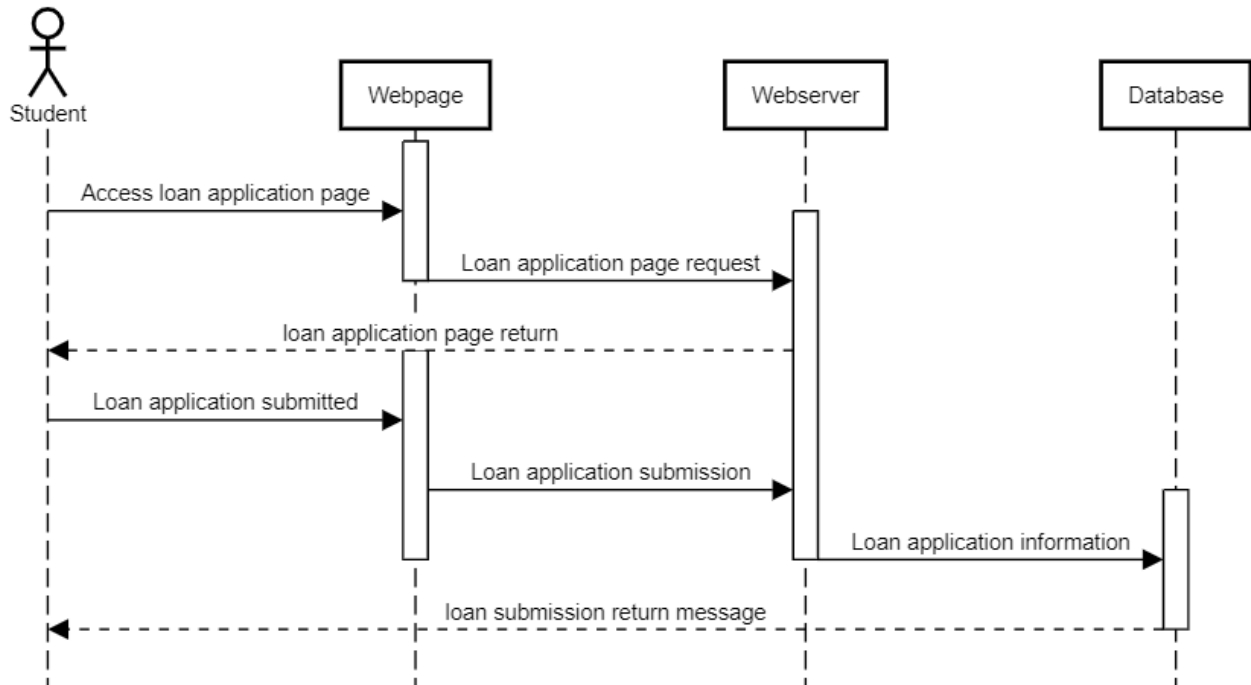


Fig. 6: The sequence diagram for loan applications

Payment redirection (Fig. 7):

A Faculty user can access the tuition billing page and request to see pending loans. On success, the Faculty will be able to request the loan balance to be redirected to the student's tuition account.

screens for the registrar to direct billing the loans directly for tuition payments

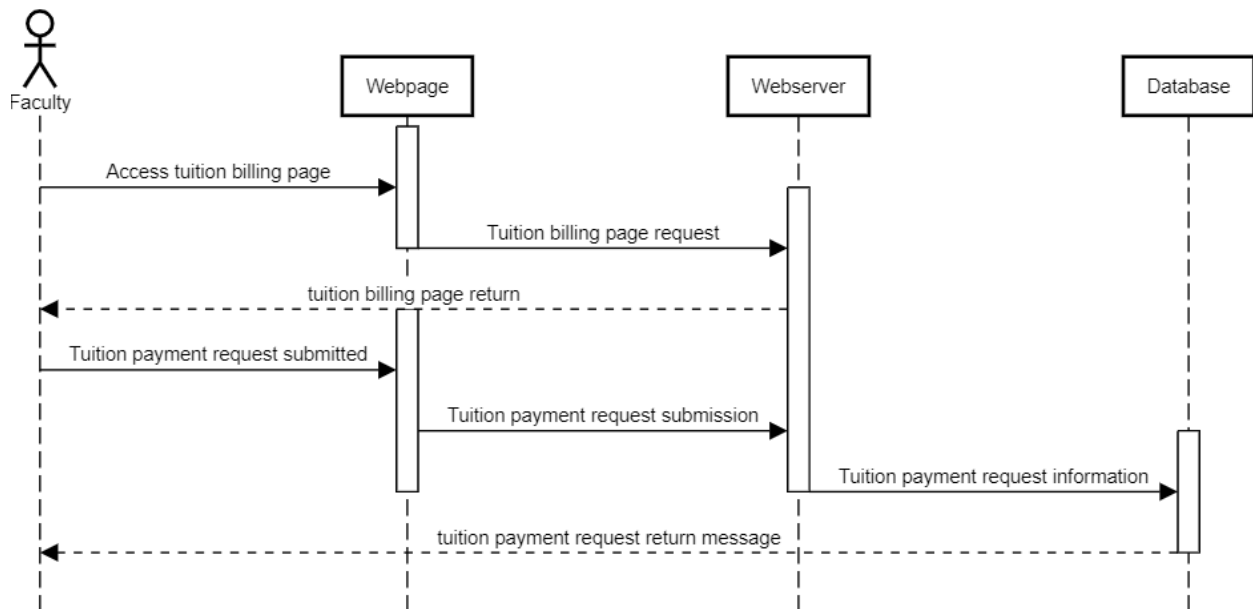


Fig. 7: The sequence diagram for tuition payment directing

State Diagram

Here we see how the user can log in and be redirected to its appropriate landing page given their userType data. They can also make an account or fail their login and be given another opportunity. From the landing pages, the User can access any of the Webpages they have access to and leverage their functionality. From any state the user can also log out and leave the site. A full state diagram can be found in Figure 8.

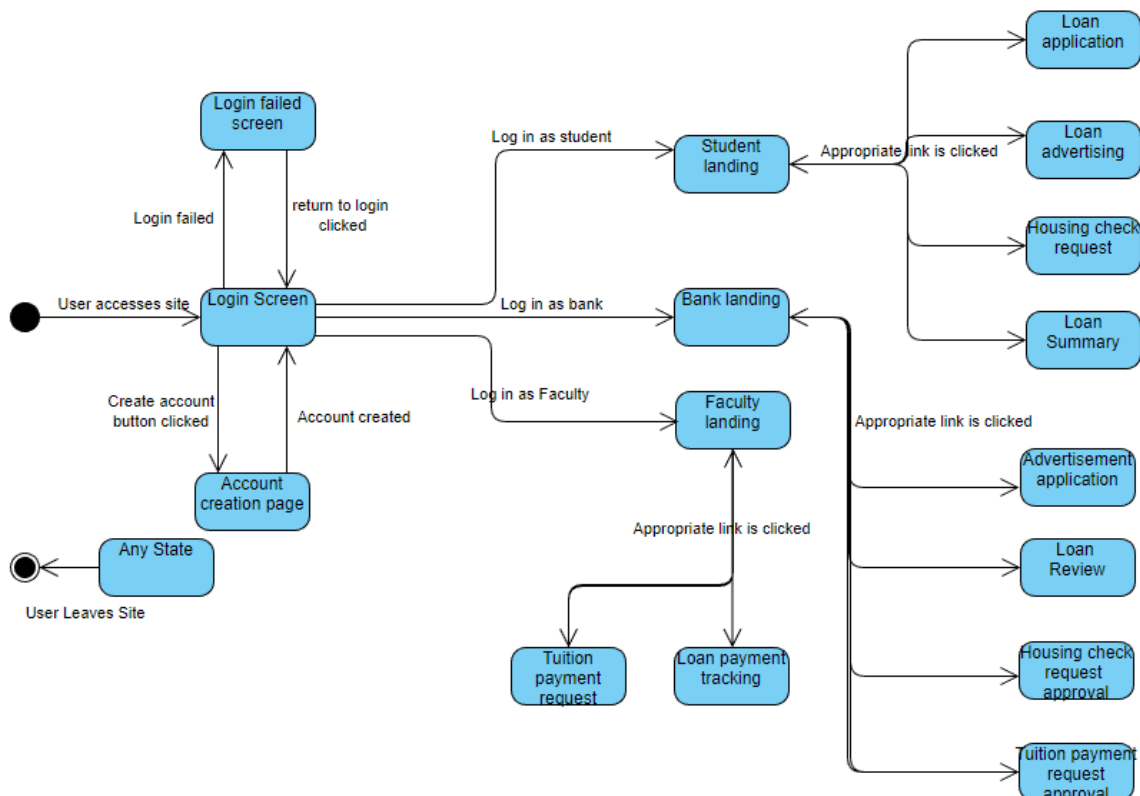


Fig. 8: The state diagram of the system webpages

Management

SCRUM Sprints Lists of User Stories/Use Cases

All user stories are added to a Jira board after being estimated for completion difficulty. This list of requirements is shown in Figure 9. Out of these, by the fifth sprint we were able to complete four key functionalities which have been described in the Sequence Chart section of this report. A list of all completed work (including supporting functionality that was developed) can be found in Figure 20.














 Form to apply for loans
 Ability to perform risk assessments on students that need a loan
 Apply for the available loans
 Student approve payment to school
 Track all student loans currently active
 Submit payment requests to the banks
 Directly pay tuition to the banks
 Financial calculator for loan values
 Create landing pages - Student, College, Banks
 Screens for loan officers to access the system to review loans
 Screens for students to access the system to apply for loans
 Interface to the database for the registrar to view who applied for loans, the status, and the amount of the loan
 Screens for the registrar to directly bill the loans for tuition payments

Fig. 9: SCRUM list of user stories as condensed titles

SCRUM Project Burndown Charts

Sprint 1:

The first sprint resolved the architecture design as well as choosing the order of features to implement. Database implementation began in this sprint but was not fully completed and rolled over to the second sprint

Sprint 1

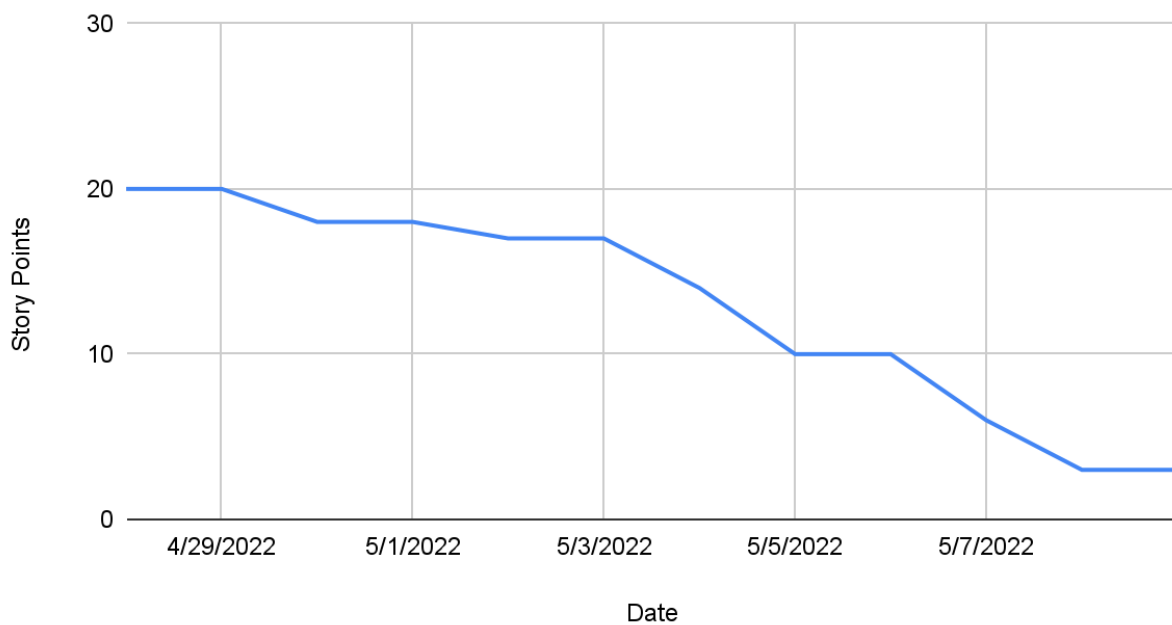


Fig. 10: SCRUM Burndown Chart for Sprint 1

Sprint 2:

Second sprint tended to be the creation of the Login page, which was completed. Database implementation to connect the Login page to the User database was also continued, but carried over to the third sprint given issues in development due to unfamiliar APIs

Sprint 2

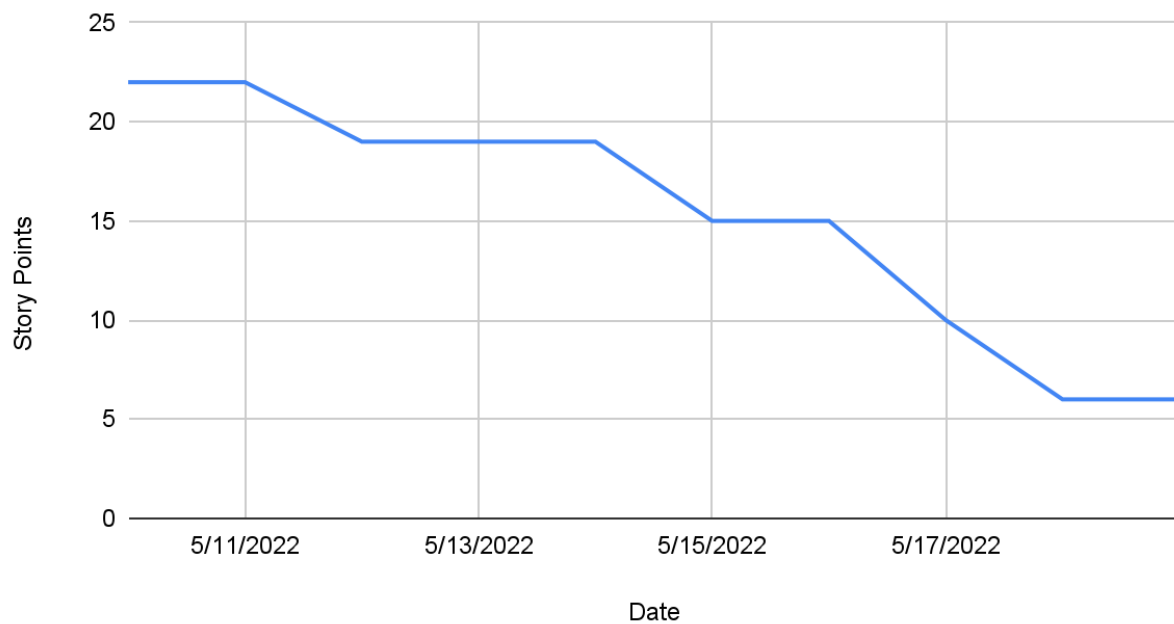


Fig. 11: SCRUM Burndown Chart for Sprint 2

Sprint 3:

Third sprint saw the final implementation of the database, which was problematic due to the unknown API. This Sprint was rather slow due to those issues. Non-sprint-related work was also performed during this sprint in obtaining permission to use the Kettering SQL server and setting up a repository

Sprint 3

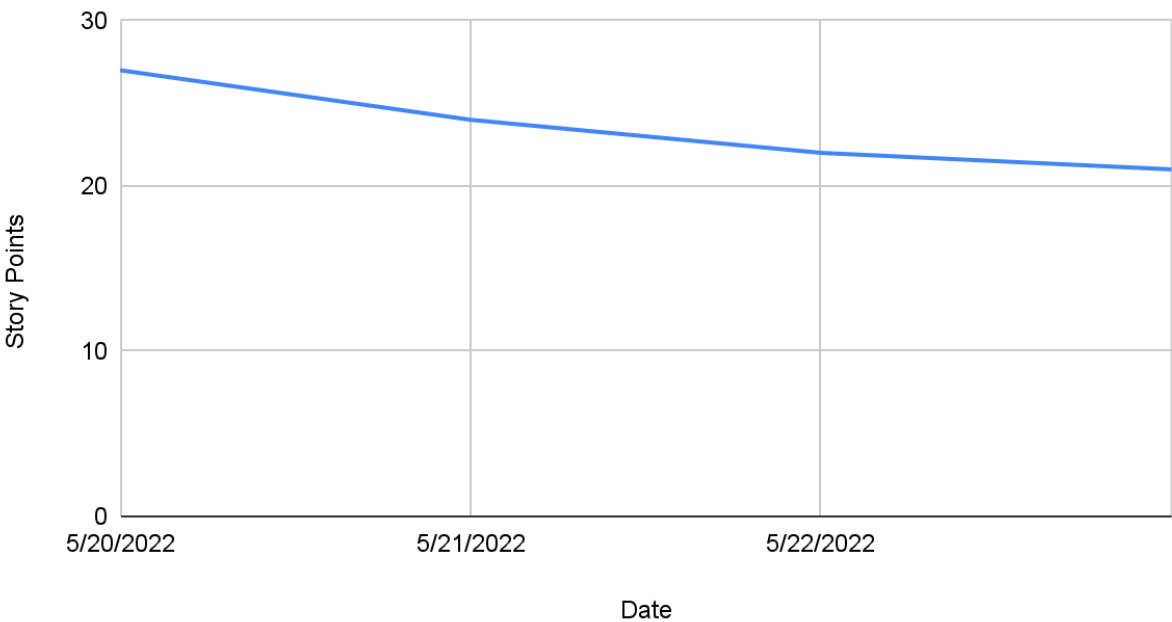


Fig. 12: SCRUM Burndown Chart for Sprint 3

Sprint 4:

The fourth sprint saw the creation of the front end components of the loan application page, the loan review page for the student, the faculty loan review page for faculty, and landing pages for all of the types of Users. Bank Loan approval page was begun but carried over to Sprint 5

Sprint 4

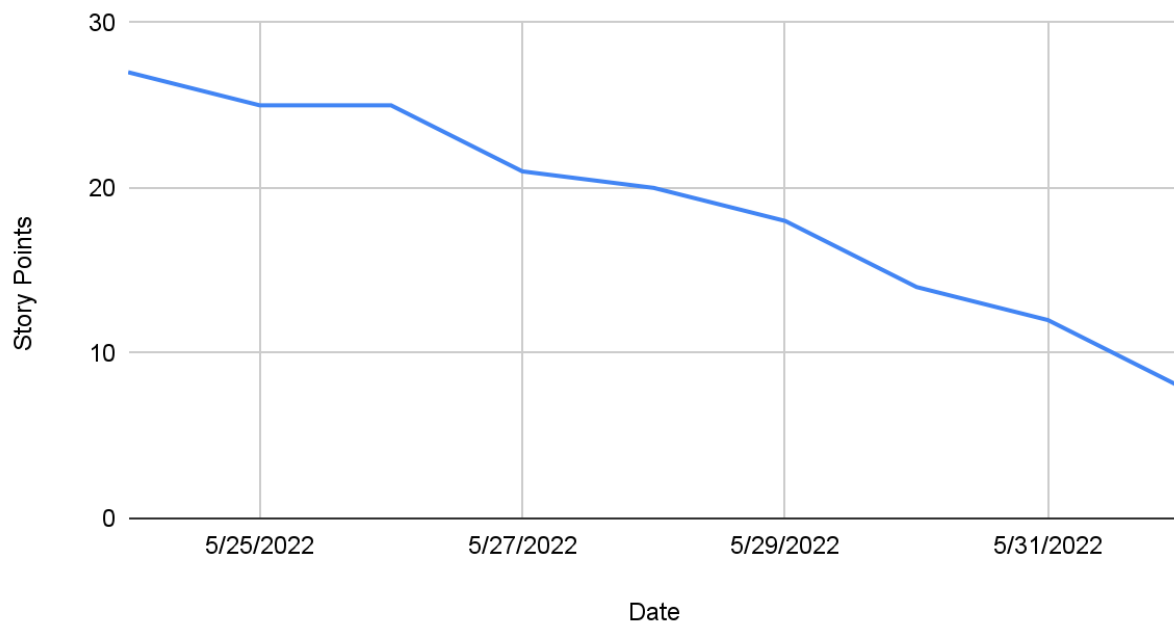


Fig. 13: SCRUM Burndown Chart for Sprint 4

Sprint 5:

This was the final sprint during our period, Loan approval, Tuition payment, and Loan visualization pages were completed for Banks. A page for Faculty to request tuition payment was added. Finally a page was added for students to approve tuition payment.

Sprint 5

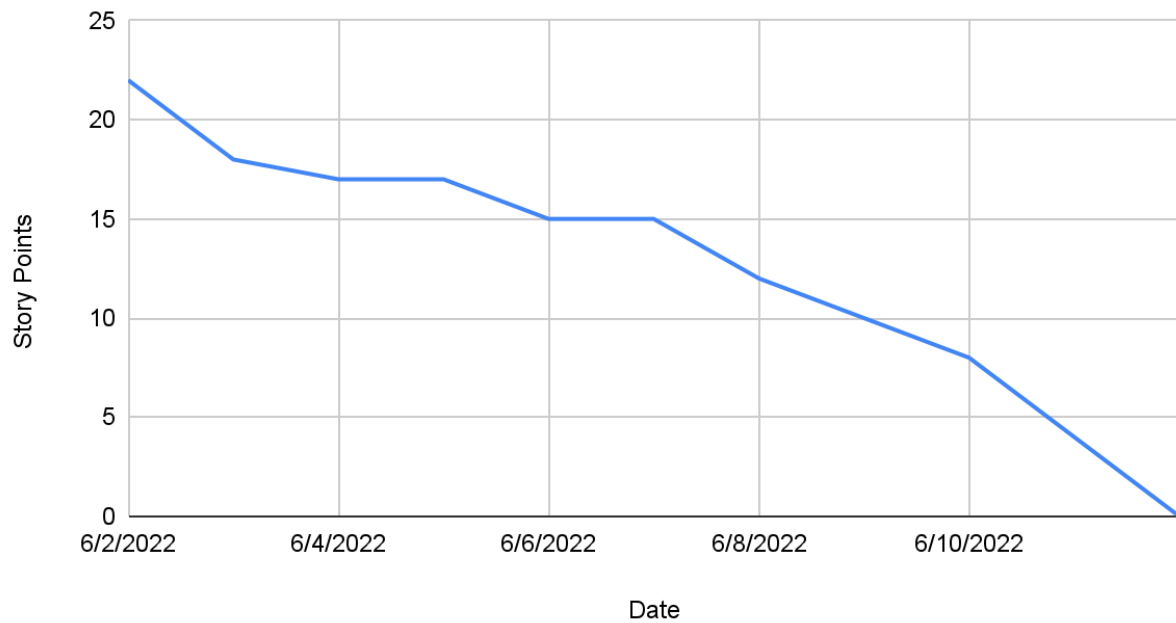


Fig. 14: SCRUM Burndown Chart for Sprint 5

SCRUM Project Velocity Chart

Our velocity chart looks as such as we were working through database issues during our first two sprints as we did not fully understand the connection and communication between our database and our application. Sprint three was associated with ironing out all of the remaining issues we were facing, and it was a shorter time period. Sprint four we started to make some progress however we were still behind our goal due to a lot of work needing to be accomplished to hit the final mark. During our fifth and final sprint we finally found the proper amount of work to distribute and was able to complete all of our planned tasks to finish the main four components we set out to complete.

Velocity Chart

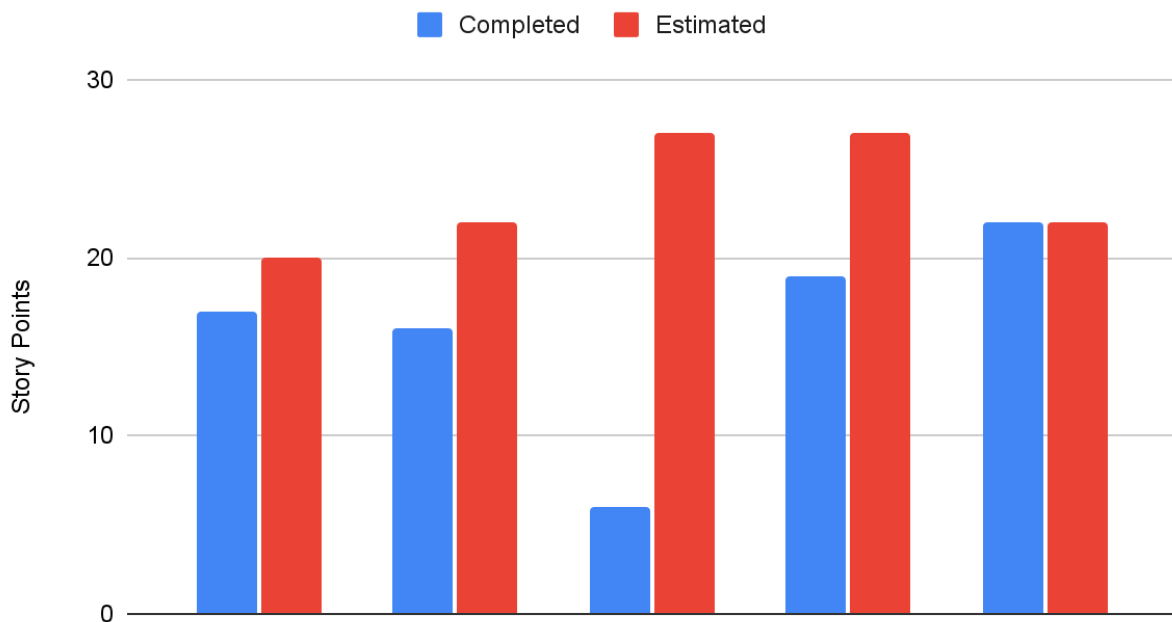


Fig. 15: SCRUM Velocity chart for each sprint

Test Cases and Test Outputs

Test cases were set up to verify the proper function of Logging in to an account (Fig. 16), reviewing/approving loans as a bank officer (Fig. 17), directing loan payments to the university as faculty(Fig. 18), and applying for a loan as a student(Fig. 19). Detailed breakdowns of each test case and their failure/success conditions can be seen below accompanied by each test's results and any relevant notes.

Project Name: Student loan system

Test Case

Test Case ID: 001

Developed by: Group 4

Test Priority (Low/Medium/High): Med

Test Executed by: Alex Gale

Test Title: Login system test

Test Execution date: 6/12/22

Description: Testing the login system

Pre-conditions: Webpage has been loaded

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Enter incorrect login information		Login information is entered into the field, password box shows dots instead	Login information is entered into the field, password box shows dots instead	Pass	
2	Press login		Login failed screen is presented	Login failed screen is presented	Pass	Login credentials were invalid, intentionally
3	Enter correct login information		Login information is entered into the field, password box shows dots instead	Login information is entered into the field, password box shows dots instead	Pass	
4	Press login		Login success screen is presented	Login success screen is presented	Pass	

Post-conditions: User has been logged in

Fig. 16: Test case for the login system

Project Name: Student loan system

Test Case

Test Case ID: 002

Developed by: Group 4

Test Priority (Low/Medium/High): Med

Test Executed by: Alex Gale

Test Title: Loan review test

Test Execution date: 6/12/22

Description: Reviewing loans as a bank officer test

Pre-conditions: Logged in as a bank officer on the bank officer landing page

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click the Loan approval page		Loan approval page is loaded, with the appropriate loan information visible	Loan approval page is loaded, with the appropriate loan information visible	Pass	
2	Approve loan		Loan approval page is presented	Loan approval page is presented	Pass	
3	View loan in loan view to check that it was entered into the system correctly		Loan is added to the system correctly, with bank approval	Loan is added to the system correctly, with bank approval	Pass	
4						

Post-conditions: New loan has been approved in the system and awaiting faculty payment fulfillment request

Fig. 17: Test case for the loan review and approval system

Project Name: Student loan system

Test Case

Test Case ID: 003

Developed by: Group 4

Test Priority (Low/Medium/High): Med

Test Executed by: Alex Gale

Test Title: Loan payment request test

Test Execution date: 6/12/22

Description: Testing the faculty's ability to direct loan payments

Pre-conditions: Logged in as faculty on the faculty landing page

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click request payment		Redirected to payment request form page	Redirected to payment request form page	Pass	
2	Select loan to request payment for		Field is filled with loan number	Field is filled with loan number	Pass	
3	Submit form		Form is submitted and success screen appears	Form is submitted and success screen appears	Pass	
4	View loan in active loans to check that it was entered into the system correctly		Loan is added to the system correctly, with payment request	Loan is added to the system correctly, with payment request	Pass	

Post-conditions: Loan payment request submitted, awaiting bank payment

Fig. 18: Test case for the tuition payment request system

Project Name: Student loan system

Test Case

Test Case ID: 004

Developed by: Group 4

Test Priority (Low/Medium/High): Med

Test Executed by: Joshua Woods

Test Title: Loan application test

Test Execution date: 6/12/22

Description: Testing to make sure you are able to apply for a loan

Pre-conditions: Logged in as a student on the student landing page.

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Click loan application form		Redirected to loan application form page	Redirected to loan application form page	Pass	
2	Ensure all fields are required		All input fields are required to make a selection before submission of the form	All input fields are required to make a selection before submission of the form	Pass	
3	Submit form		Form will submit and confirmation of form submission screen comes up	Form will submit and confirmation of form submission screen comes up	Pass	
4	Return to student landing page		Student landing page link works as expected	Student landing page link works as expected	Pass	
5	Click track active loans		Redirected to a new page that shows all loans applied for including the one that was just applied for	Redirected to a new page that shows all loans applied for including the one that was just applied for	Pass	

Post-conditions: New loans are added and awaiting bank approval

Fig. 19: Test case for the loan application system

SCRUM Final List of Work Completed

In Jira, our final list of work completed shows all of the work that we completed. A lot of our tasks were sub tasks of the ones in this list proving to be much more work than these ones here alone. However these are the main points during our project that we got through and took care of.











	CS417-13
Architecture design	
	CS417-26
Fully web based access for 3 typ...	
	CS417-7
Ability to request checks monthly	
	CS417-3
Advertising section with banks av...	
	CS417-25
Bank advertisement page	
	CS417-40
Create landing page to hold all of ...	
	CS417-31
Monthly Housing Check request	
	CS417-14
Choose features to implement	
	CS417-21
Create frontend for the login scre...	
	CS417-1
Create Database	

Fig. 20: SCRUM Final list of main task work completed

Code Written

We developed code during the sprints mainly using PHP to interact with the database and to keep session variables. Our database holds all of the information on the users, and the loans that the users are applying for. The user data is stored in its own table, and is used solely to track the users on the application consisting of a ID, username, password, user type, and a bank name field if applicable. On creation of user accounts they are stored in the database and then used for verification during login. On login they are presented with their basic account information such as their ID, their username, password, and user type. There will be a link to their respective user type landing page, student, bank, or university. If the user is a bank, on first login a second link will also be displayed, that link will take them to a page where they declare which bank they are a part of and will be able to view the loans of.

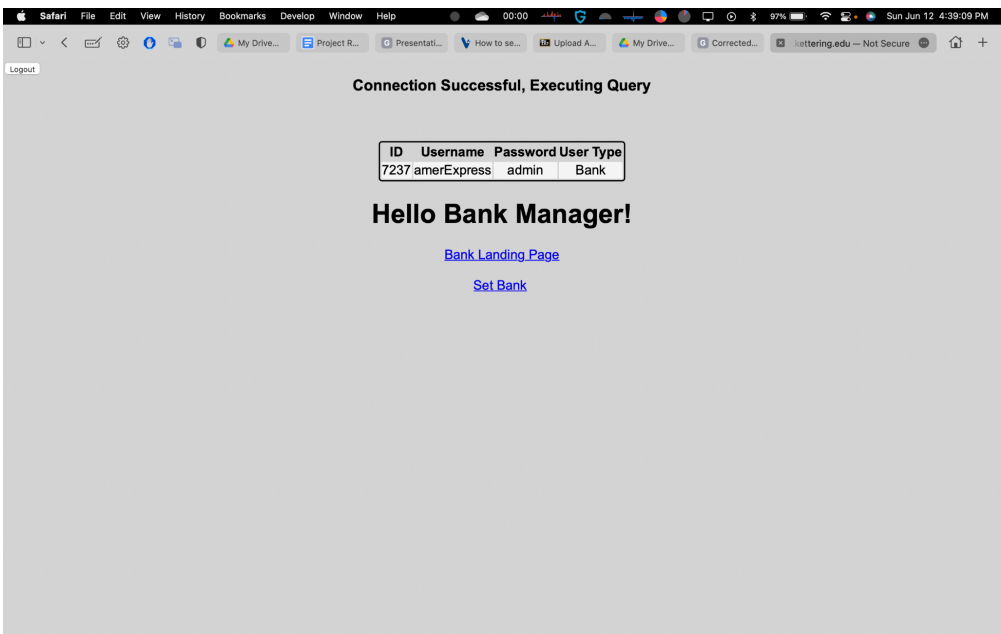


Figure 21: Bank Login

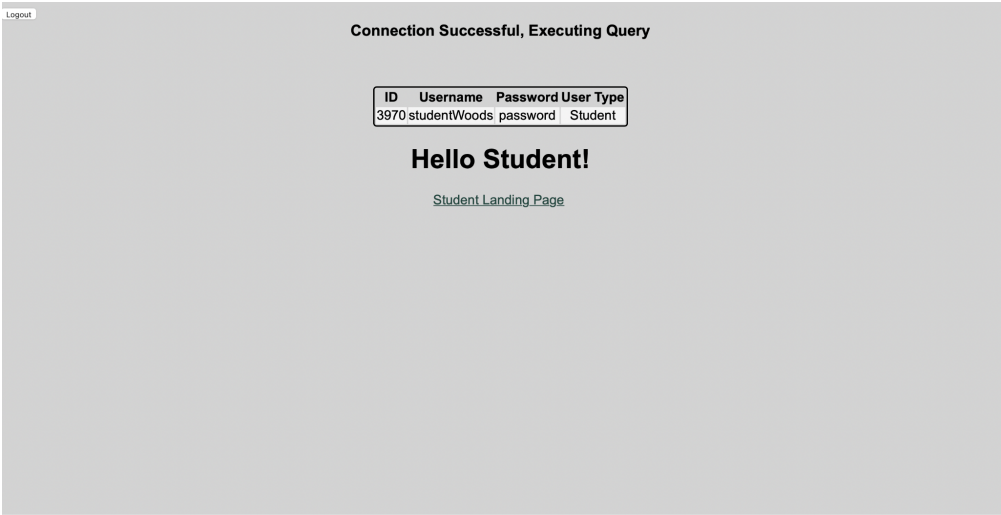


Figure 22: Student Login

The loan application table holds the username of the person applying, the bank that is distributing the loan, and more basic information about the loan such as the date, the rate of the loan, and more.

Connection Successful, Executing Query

Username	Distributing Bank	Applied Date	Loan Amount	Credit Score	Approval Status	Loan Rate	Loan ID	Amount Paid	Payment Status
studentWoods	American Express	2022-06-12	\$1000	675	Approved	16%	809	\$0	Approved by Student
jwoods	Chase	2022-06-09	\$10000	100	Applied	0%	559	\$0	
jwoods	Capital One	2022-06-08	\$1000	600	Applied	0%	4114	\$1	
jwoods	Chase	2022-06-08	\$100	10	Approved	12%	877	\$100	Completed
jwoods	American Express	2022-06-08	\$1000	100	Applied	0%	737	\$0	Approved by Student
jwoods	Chase	2022-06-09	\$1000	650	Approved	13%	652	\$1000	Completed
jwoods	Chase	2022-06-09	\$1000	100	Approved	16%	166	\$0	Awaiting Bank Payment
jwoods	Huntington	2022-06-09	\$1000	100	Applied	0%	189	\$0	
wood92	American Express	2022-06-09	\$10000	800	Applied	0%	932	\$0	

University Landing Page

Figure 23: Database Display

As a student, they have the ability to apply for loans, and have to wait to see if the bank then approves or denies their application. This is done by the student viewing their active loans that only display the loans associated with their account. There is a column under the name of “Approval Status” that can be “Applied”, “Approved”, or “Denied”. “Applied” is automatically set once the user applies for their loan, and the Bank has the option whether to approve or deny the loan. Once the bank approves the loan, it is then back on the student to approve payment to the University. This is to ensure that the student knows that the loan is approved and that they are ready for the school to request their tuition payments from the bank. Once the student approves the loan, the Payment status updates to be approved by the student, so the University knows that the student is ready for payments to start. From the University landing page, the University administrator can either view all of the loans that have been applied for, or they can view the loans that are ready for payment to be requested. For the University to approve a loan all they need to do is enter the loan ID of the loan that is ready for approval.

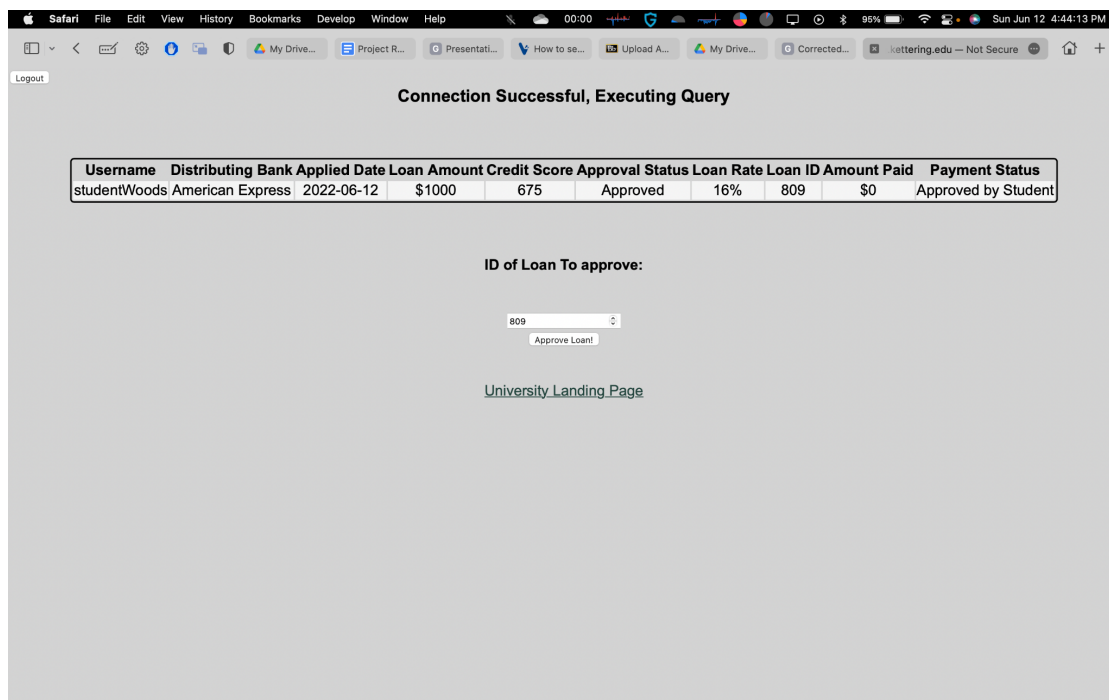


Figure 24: University Approval

Once the loan is approved, by the University the payment status column updates to “Awaiting Bank Payment” and stays that way until the loan is paid off completely. Once the loan is completed the payment status column auto updates to “Completed” through a SQL query that updates the value if the amount paid and loan amount are equal.

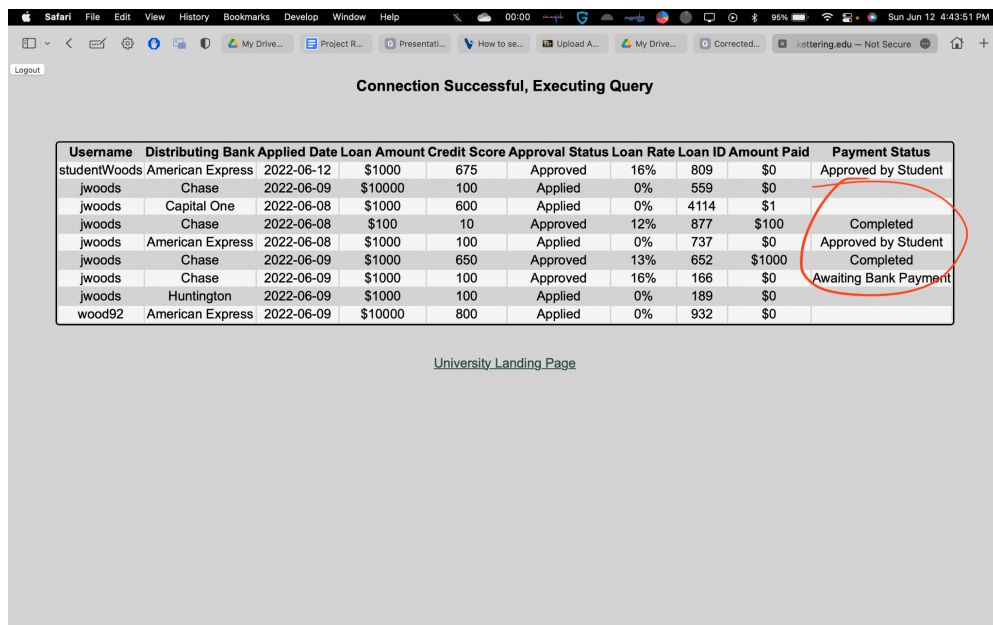


Figure 25: Completed Payment