# **CSC289 Programming Capstone Project**

**Project Name:** Integrated Library System

**Team Number:** 7

**Team Project Manager:** Denitri Douglas

**Team Members:** Phoebe Andrew, Connor Nelson

**Mentor:** Lisa Schenkewitz

**Release Report**

**Integrated Library System- Version 1.0**

**Overview**

This library management software is designed to handle various tasks related to a library, including managing books, patrons, librarians, and administrators. The system uses a MySQL database to store information.

**Development Highlights**

* **Project Initiation:**

Project Kickoff: The kickoff meeting was held on January 19th, 2024, discussing the SRS document provided and determining the scope of the project. The syllabus was reviewed to establish the timeline, affirming the achievability of the provided SRS without alterations despite group size.

* **Requirements Gathering:**

During the project scope review, discussions were held to select the optimal tools for software development. Python was chosen as the programming language due to its extensive libraries and support. Library Data is being stored in a MariaDB database. AWS (Amazon Web Services) is used for website hosting, employing EC2 and S3 storage and Nginx.

Django was preferred over flask for its database data manipulation ease.

An Agile board was created on Trello to manage user stories and maintain

development within scope each sprint.

* **Design and Architecture:**

Several meetings were conducted to outline the class structure for the Python program and the database schema. Modifications were made along the way concerning fee management in the program, yet the initial design remains unchanged.

* **Development Progress:**

The project was developed over five two-week sprints, with biweekly meetings and daily project discussions via Teams chat. These meetings encompassed sprint reviews and retrospectives to address bugs, solutions, and potential design changes. Weekly revisits to user stories were performed as the project evolved.

* **Testing and Quality Assurance:**

Test Types: Unit, Integration, Functional

PyCharm debugger used during development

Function Tests:

get\_book\_description: Checks if book descriptions are retrieved properly.

calculate\_fee: Ensures correct calculation of overdue book fees.

* View Tests:

checked\_out\_books\_view: Verifies successful loading of checked-out books page.

checkin\_view: Tests book check-in functionality.

create\_account\_view: Validates user account creation page loading.

Result: Ensures proper functioning of key features in the library application.

* **Bug Fixes and Enhancements:**

The most notable fix concerns the fee handling system. Initially, fees were stored in the book table and cleared upon book check-in. Now, fees are stored in a separate fee table, and they are not automatically cleared upon book return.

**Deployment**

* **Launch an EC2 instance**: Chose an appropriate instance type for our project and launched it on AWS.
* **Set up database**: Configure and make the database accessible from the EC2 instance.
* **Install Nginx and Ggnicorn**: Use SSH to connect to the EC2 instance and install Nginx and Gunicorn.
* **Configrure Gunicorn**: Create a gunicorn.service and gunicorn.socket file as described in the documentation
* **Configure Nginx**: Set up Nginx to serve the Django application by creating a nginx.conf configuration file.
* **Set up Python environment**: Install Python and dependencies for the Django app on the EC2 instance.
* **Deploy Django app**: Transfer the Django application code to the EC2 instance (preferably using git).
* **Configure Django settings**: Update Django settings in the config.json file for the production environment.
* **Collect static files**: Use Django's collectstatic command to gather static files.
* **Restart Nginx**: Applied the configuration changes by restarting Nginx.
* **Test website**: Accessed the Django website via the EC2 instance's IP address or domain name and ensure everything functions correctly.

**Release Notes**

* **New Features:**

The book of the month is displayed as the main page when interacting with the website. The books are randomized, and the book descriptions are pull via title from the google books API.

* **Bug Fixes:**
  + The fee system has been fixed and enhanced using modal to simulate the “payment” of fees.
  + User authentication is being handled by Django, as the framework offered a more secure solution.
* **Known Issues:**

The database schema was loaded with ISBNs from amazon, and they do not match with those on the google books API. Therefore, the book of the month page provides descriptions based on book title instead of ISBN.

**Conclusion**

A Python-based solution leveraging Django and AWS was successfully implemented to create an efficient library management system. An Agile approach, coupled with daily communication and sprint reviews, ensured timely progress and quality development.

Version 1.0 establishes a robust foundation for the system with new features like the "book of the day" display and improved fee handling. Future iterations will address known issues, particularly ISBN data discrepancies, and refine the system based on feedback.