University of Arkansas – CSCE Department
Capstone I – Final Proposal – Fall 2022

Data Science Room Scheduler

Alexandria Lim, Wesley Parker, Miguel Campbell, Jackson DiRienzo, Creighton France

Abstract

The University of Arkansas Data Science Program would like a solution that will allow Data Science students to reserve rooms in the new Champions Hall common space. Having a common space solely for the Data Science Program fosters a sense of community and collaboration amongst the department's students, so it is important that this space be efficiently used. The capstone team will provide a solution by building a web application in Python, which students can use to reserve rooms for maximum usage efficiency, and it is an opportunity for Data Science students to be involved in the application's future development and maintenance.

1.0 Problem

Data science is a field that is currently growing in popularity and relevance to our rapidly digitizing society, and there is a high need for data scientists. The recently founded Data Science Program at the University of Arkansas just received a new common project space in Champions Hall. Having a dedicated space in which to work and meet is not only important for collaboration on group projects but is also helpful for fostering a sense of camaraderie and community between students in the same program.

Not having an organized, centralized way to share the common space can lead to many unfavorable scenarios. There might be many students wanting to use a room, or sometimes none. A few students may use a room for an extended period and other students would waste time waiting for a room to become available. Conversely, students can be unaware that rooms are available, and the rooms are not used at all, which is a waste of the rooms. Ultimately, this lack of a way to coordinate the use of the Champions Hall common space could lead to students not using the rooms at all, because they are not sure if a room will be available. Leaving the rooms without a method to ensure students can use a room is not an ideal or effective way to share the space meant for all students.

2.0 Objective

This project's objective is to create a scheduler application that gives all Data Science students the opportunity to reserve rooms in Champions Hall for collaboration on group projects and

studying. Students should be able to survey a room's availability, invite other students to meetings, book new reservations, update existing reservations, and cancel reservations if needed.

3.0 Background

3.1 Key Concepts

There are a variety of technologies that will be used in the web application. At the core of these technologies is the Django web framework. Django is Python-based and is ideal for data-dependent web applications due to its simplicity [1]. Django's all-around use of Python ensures consistency and familiarity for the Data Science students, who are expected to take on future development and maintenance. Administrators will also benefit from Django's admin site, an administrative interface that allows authorized users to manage the application's content [2].

For the database, we will be using SQLite. SQLite is included with Django, so there are no additional installations needed. SQLite uses the computer as the host so that data can be freely edited at any time. This will allow the site to store information so that users can log in and the admins can monitor and make edits to different variables on the site. It will store the availability and information of rooms so that students can know what times they can reserve. All this data can be edited using Django code without having to directly access the database itself.

Another key technology is the Outlook calendar. Our web application will send Outlook emails to students and faculty containing ICS Calendar files of their upcoming reservation, which can be added to Outlook calendars. Because all students and faculty are given an Outlook account, everyone will be able to update their calendars. This is to help ensure that scheduled rooms are constantly occupied.

We will be utilizing the University's Single-Sign-On (SSO) service to secure our application. SSO is an authentication method that permits a user to utilize one set of login credentials to gain access into multiple applications [4]. Since we are using the University's system, this allows students and faculty to employ their own university credentials to access our application. Thus, no new credentials need to be made, and all the firewalls and security the university has in place will also protect our application.

3.2 Related Work

This Data Science room scheduler is not the first of its kind. Many applications and libraries can accomplish the same set of tasks we wish to do in our application. For example, the University of Arkansas uses a scheduler for study rooms at Mullins Library [3]. The Mullins scheduler provides an appropriate user interface which will be useful as a reference for our own views. However, the Mullins scheduler can be used by all university students, and we need to ensure that our scheduler is only accessible to Data Science students.

The University of Arkansas's Department of Theatre also utilizes its own room scheduler for theatre practice [4]. Like the Data Science Program, the Department of Theatre only has a limited selection of rooms to reserve. The user interface used for this scheduler shows a Microsoft calendar widget, which can be connected to Microsoft Teams and Outlook like the calendar notification we would like our application to generate. However, a student does not

book a reservation through the application – instead, the student finds availability through the application, and then emails the department a request.

Another application we will reference is ReZerve. Originally developed by a CSCE Capstone team during the Fall/Spring 2020-2021 year, ReZerve is a web application designed to handle appointments for barber shops and stylists. The business logic behind ReZerve's appointment scheduling components will be a helpful resource when implementing a similar component on the room scheduler. But ReZerve is written in TypeScript using React's framework [5], whereas our application will be written in Python using Django's framework.

4.0 Design

4.1 Requirements and Design Goals

When deployed, the application must satisfy the following requirements:

1. Accessing the application should not require a download of extra software (ideally the application should be web-based with a desktop-oriented design).
2. The programming language of the application must be one Data Science students are proficient in so that they can provide maintenance and advancement of the application in the future.
3. The application should only be used exclusively by students in the Data Science Program or authorized guests.

4.2 High Level Architecture

Django uses MVC (Model View Controller) architecture. We have a design of models, views, and frontend mockups, which are outlined as follows:

Views

View	Endpoint	Description
Login	/login	- SSO or external login. SSO is done by OIDC login through the university's Microsoft Azure system. External login is done with the classic Django login functionality.
Admin Dashboard	/dashboard/admin	- Admin view endpoint. Admins can see all reservations, users, and reports along with the ability to accept pending reservations or to create unavailable times.
Dashboard	/dashboard	- Shows user information, reservations for the current user, can cancel current reservations (button – "Cancel"), can make a new reservation (button – "Reserve")
Reserve a Room	/reserve	- Shows selected availability, users can invite members of the reservation, and give the reason for the reservation

Confirm Reservation	/reserve/confirm	- User must agree to the code of conduct, and confirm - After doing so, their reservation will be added to the database and Outlook notifications will be sent to all team members
View a Reservation	/reservation/{id}	- User can view information about a specific reservation if they are the creator, an admin, or a member that was added to the reservation. - Admins and reservation creators will be able to go back to their dashboard from this view, while those simply viewing the reservation from an external link will be presented with a 'view-only' layout.
Cancel Reservation	/reservation/cancel/{id}	- User is prompted with information on their reservation and asked to confirm that they would like to cancel it. - After doing so, their reservation will be marked as 'cancelled' and each participant will receive an email stating their reservation has been cancelled.

Models

Users

Field	Django Type	NULL?	Default	FK	Notes
id *(PK)*	IntegerField	NOTNULL			
email	EmailField	NOTNULL	SSOS		From SSO on creation (UARK accounts)
group	IntegerField				From SSO on creation (UARK accounts), student, faculty, staff, other
first_name	CharField	NOTNULL			From SSO on creation (UARK accounts)
last_name	CharField	NOTNULL			From SSO on creation (UARK accounts)
can_book	BooleanField		True		Used to ban users from booking

Rooms

Field	Django Type	NULL?	Default	FK	Notes
id *(PK)*	IntegerField	NOTNULL			
name	CharField	NOTNULL			Room name
number	CharField	NOTNULL			Room number (such as 123B)
min_capacity	IntegerField				
max_capacity	IntegerField				
is_reservable	BooleanField		True		If room is unavailable entirely, set this to False
needs_approval	BooleanField		false		If true, reservations booked with this room is not automatically approved

Reservations

Field	Django Type	NULL?	Default	FK	Notes
id *(PK)*	IntegerField	NOTNULL			
user_id	IntegerField	NOTNULL		Users:id	
room_id	IntegerField	NOTNULL		Rooms:id	
start	DateTimeField	NOTNULL			
end	DateTimeField	NOTNULL			
note	TextField				
status	BooleanField	NOTNULL	0		PENDING = 0 APPROVED = 1 CANCELLED = 2 REJECTED = 3

Reservations Participants

Field	Django Type	NULL?	Default	FK	Notes

id *(PK)*	IntegerField	NOTNULL			
user	IntegerField	NOTNULL		Users:id	
reservation	IntegerField	NOTNULL		Reservations:id	

Availability

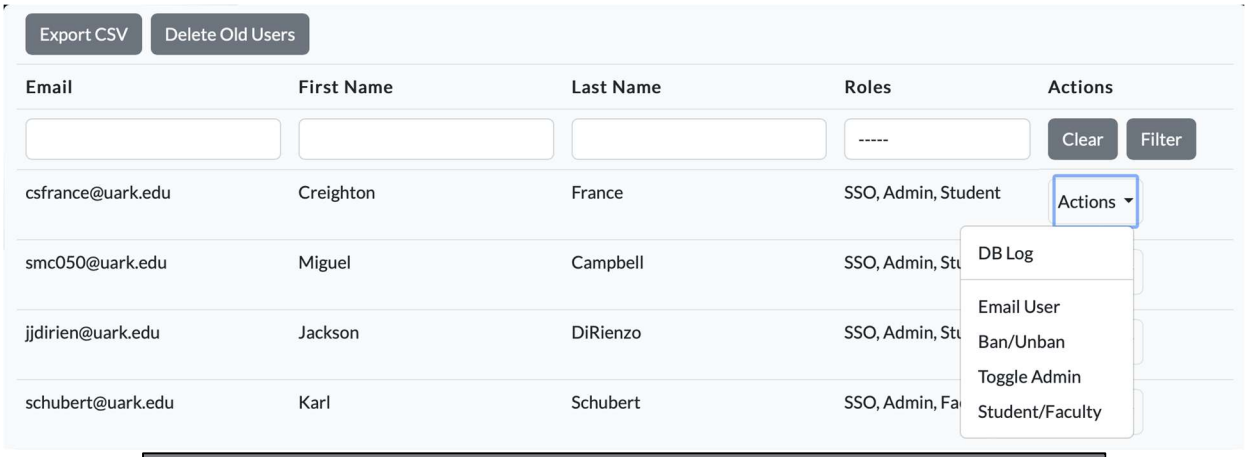
Field	**Django Type**	**NULL?**	**Default**	**FK**	**Notes**
id *(PK)*	IntegerField	NOTNULL			
type	CharField	NOTNULL			
value	CharField				
period_start	DateField				
period_end	DateField				
available	BooleanField	NOTNULL	True		
start_time	TimeField	NOTNULL			
end_time	TimeField	NOTNULL			
enforced	BooleanField	NOTNULL	True		

Report

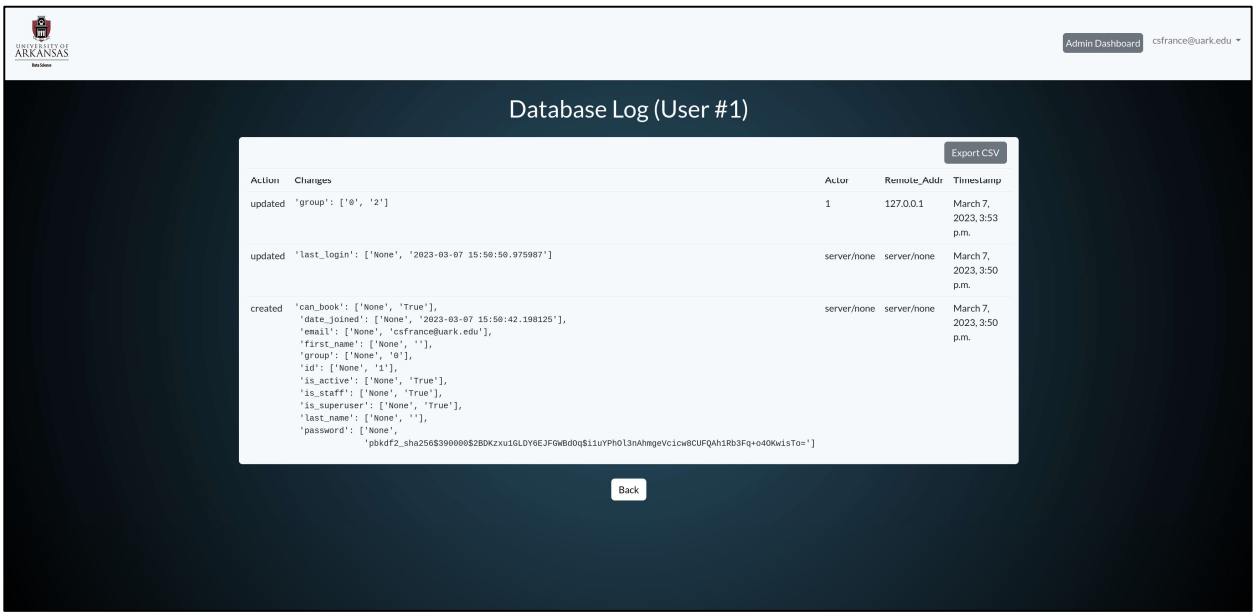
Field	**Django Type**	**NULL?**	**Default**	**FK**	**Notes**
id *(PK)*	IntegerField	NOTNULL			
message	TextField	NOTNULL			
user	IntegerField	NOTNULL		user:id	
reportDate	DateField	NOTNULL			
viewed	BooleanField	NOTNULL	False		

Views

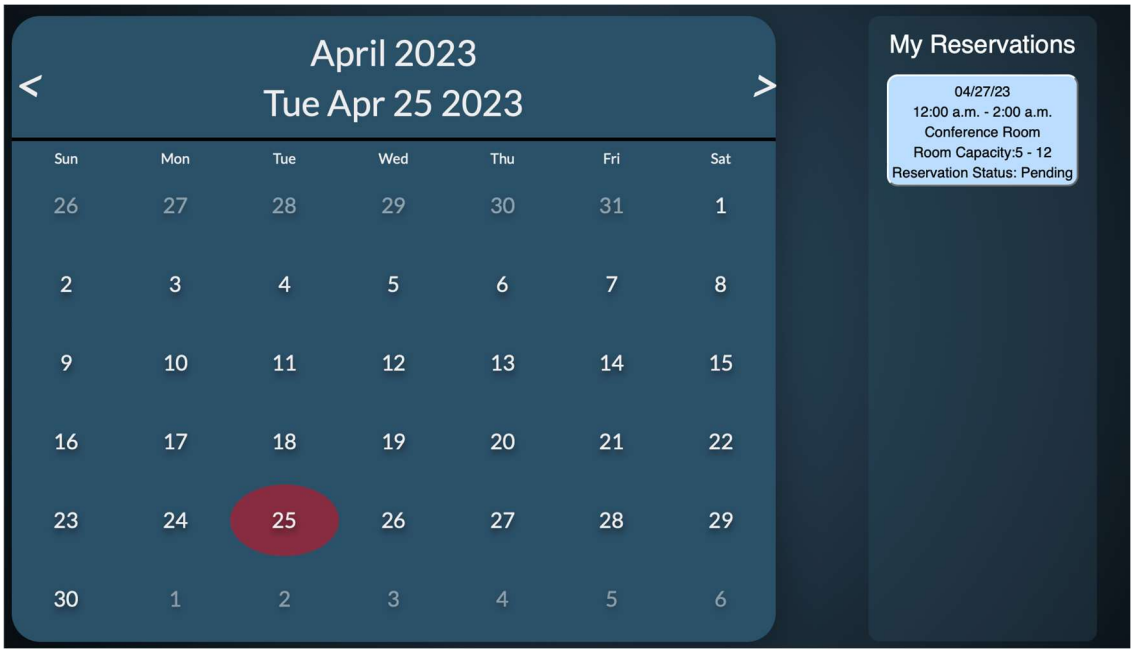
Admin Manage Grid



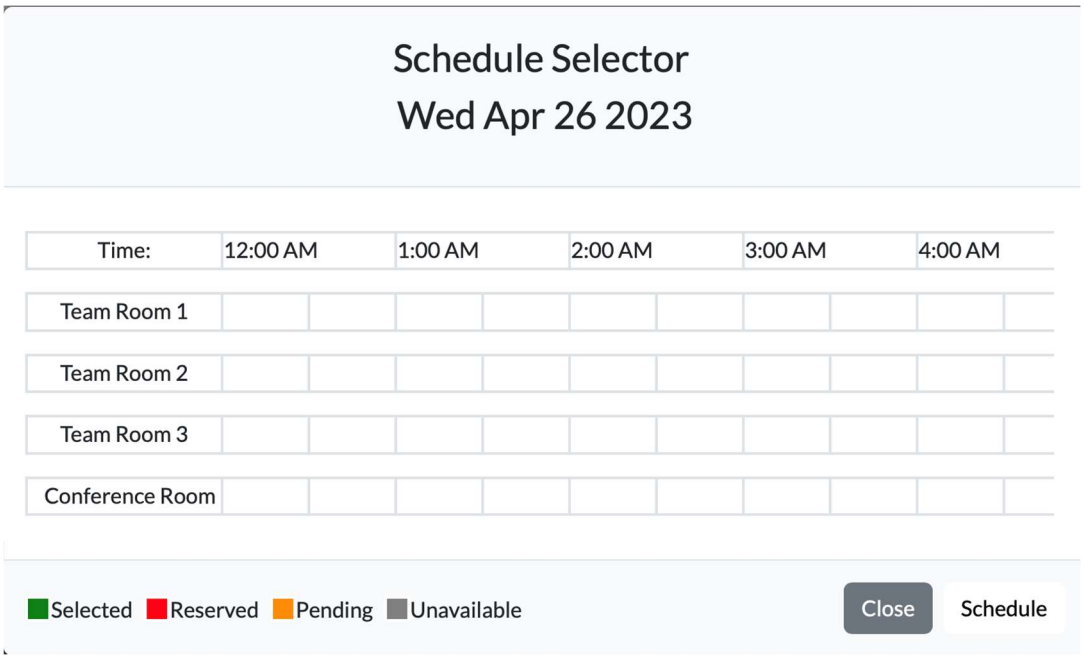
Admin DB Logs



Dashboard



Dashboard – Overlay



4.3 Low Level Design

Login: The application has two different methods of logging in – one for authorized external guests that are not affiliated with the university, and one for the Data Science students and faculty within the university. The external guests are authenticated with the Django login functionality, and the Data Science students and faculty are authenticated with the UARK OIDC login functionality.

While the application uses the basic Django login functionality, it is customized to fit the specific needs of the application. The custom User model for the room scheduler application uses the email address as the username and adds the fields group type (students, admin, or external), first name, last name, and a Boolean value that shows if they can book. The custom User model also needs its own "UserManager" class to create these users for the database. Furthermore, a custom backend authentication is required to authenticate users based on their email field instead of their username field, which would no longer exist under the custom User model.

The other form of authentication is OIDC login through the university. The Python library used is MSAL, which is the Python identity management library for Microsoft, and PyYAML, which parses Python YAML configuration files. In the Microsoft Azure app registration portal, the necessary application ID, secret, redirect URIs, scopes, and authority URI are assigned and added to a configuration file in the application. Then, the application can interface with the Microsoft Graph API.

Dashboard: The dashboard webpage serves two primary functions: allowing users to create study room reservations and displaying their current scheduled appointments. The room reservation function is designed around a user-friendly and visually appealing calendar. Users can select their preferred date which will then pop up an overlay that will facilitate their time and room selection. After making their selection, users are redirected to a confirmation page where they will be allowed to check that all the information up to this point is correct.

The calendar's appearance is implemented in Bootstrap, HTML, and CSS. Particularly, a grid system is used to align the webpage. The dates shown and the calendar's ability to change months is accomplished all with JavaScript. JavaScript handles the rendering of the calendar and facilitates the interaction when the user selects a date. Once a date is selected a bootstrap model appears on the screen that allows the user to schedule and submit their required time. Time selection is handled in 30-minute blocks and only one room may be specified at a single time. Because there will be multiple users using the website, an AJAX request is sent to our Django framework and all reservations for the specified date are transferred from the SQL database back to the front-end. All-time blocks that have been reserved in the database are then shown as reserved on the screen. This is all implemented using JavaScript and jQuery as the front-end languages and Python as the back-end language.

Next to the calendar, there is a dedicated area for users to view their scheduled appointments. This area displays all appointments in a list format and is constantly updated from the database. Users can click on any of their appointments to be redirected to a webpage that shows them all the relevant information about their reservation, including the date, time, room number, and any additional notes.

Overall, the dashboard webpage provides a streamlined and efficient solution for scheduling study rooms. With its user-friendly design and easy-to-use interface, users can easily reserve a study room and keep track of their appointments in one convenient location.

Reserve: When the user reaches the reserve endpoint, they are presented with a web form. By default, the webpage populates the name and email fields with the appropriate information attached to that user's profile. Furthermore, the desired room and reservation times are also

populated based on the user's input from the dashboard calendar. The user is presented with a field where they may include the emails of the participants they wish to add to the reservation. This occurs on one textfield, and emails must be separated by commas. The user also has another field where they input a "reservation note" to provide additional context about the reservation. The user may then click one of two buttons. The "Back" button takes the user back to the dashboard without saving any changes for the reservation, and the "Submit" button sends the user to the Reserve Confirmation endpoint, where they will confirm the appointment.

Reserve cancel: When a user reaches the cancel endpoint, they are provided with the information on the current reservation. Underneath the information panel, the user has two options. Clicking the "Back" button sends the user back to the dashboard without any further action. However, clicking "Cancel Reservation" sets the reservation's status code is set to "2".

The status of the reservation is stored within the reservation object under the "status" attribute. This attribute holds an integer that is mapped to a constant. The current constants are:

- When the status of a reservation is 0, it means that the reservation is pending approval by the admins.
- When a reservation's status is 1, it means the reservation was approved by the admins.
- When the status of a reservation is 2, it means that the reservation has been cancelled by the user.
- When the status of a reservation is 3, it means that the reservation has been rejected by the admins.
- When the status of a reservation is 4, it means that the reservation is a buffer defined by the admins to make certain times unavailable.

When a reservation is instantiated, its default status is set to 0. This function exists primarily for the conference room, which our sponsors must approve before a reservation is booked. An email will not be sent for this approval, instead, the admin webpage will show that a pending request has been received. When a room other than the conference room is booked, the status is automatically set to "Approved," and that status is only changed if a user elects to cancel that appointment.

Email: Django makes it easy to set up email functionality for their web applications. In the settings.py file of the application, the email backend (the basic Django core mail backend), the email host (SMTP for Office 365), the email port that the email will be sent from, and the email host user (an email service account that will only be used to send messages and not receive) are defined. A service account has been provisioned by UARK IT for the email address dascroom@uark.edu, which is the previously mentioned email host user. All email correspondence will be sent from this address.

Django also provides classes that can craft and send email messages. When a reservation is accepted, an ICS file containing the date and times is automatically generated using a Python library. This ICS file is attached to an email message and sent to the email addresses provided in the participants list for each reservation upon confirmation of the reservation. For instances where a reservation is pending, denied, or cancelled, then an appropriate email message is sent to the email addresses of all participants without an ICS file. Upon confirming a reservation, the emails will be sent by dascroom@uark.edu and sent to all participants. Replies to these emails will not be answered.

Admin Manage Grids: The management grids allow admins to view users and reservations as well as provide them with numerous actions to perform on specific objects such as deleting and altering fields. Each action is defined in our 'actions' module and they are assigned specific URLs. When a POST request is made to one of these URLs, the database is modified and a redirect to the original grid is performed. These are especially useful when the admins would like to ban users from booking, delete old students, export to CSV, or approve reservations.

Admin DB Logs: The DB Log view allows the admins to view every modification made to a specific object in the database. This is particularly useful when an issue in the application is found, and the admin needs to trace the history of an object. We are taking advantage of the Django-AuditLog Python plugin. This plugin creates an audit log table that keeps details on every database transaction. When an update is made to User 1, the 'history' field of User 1 is appended to contain the ID of the newly created audit log entry. This allows the history of a DB entry to be quickly rendered.

Report: The report view allows users to report issues with the rooms and any issues with the site. Once they fill out the report message box and hit the submit button, the application makes an entry in the database that contains the message, the time of submission, and the user ID. It also states whether an admin has viewed the report or not.

4.4 Risks

Risk	Risk Reduction
Security	The application will integrate with the pre-existing Single-Sign-On configuration. The authentication portion of any application is a point of risk, but the SSO will be maintained by the university.
Maintaining it after we leave	To allow Data Science students to easily maintain the application, we are developing it in Python, a programming language that Data Science students are familiar with.

4.5 Tasks

Our team uses Trello to organize our tasks. The following tasks and associated subtasks will be completed by the end of the Spring 2023 semester:

1. Complete design document
   - **1.1** Finalize model structure
   - **1.2** Finalize views
   - **1.3** Finalize front-end design mockups
   - **1.4** Outline
2. Complete final project proposal/presentation
3. Code up website
   - **3.1** Implement models
     - 3.1.1 Users
     - 3.1.2 Admins

- 3.1.3 Rooms
- 3.1.4 Reservations
- 3.1.5 Availability

3.2 Implement views

- 3.2.1 Login
- 3.2.2 Logout
- 3.2.3 Admin
- 3.2.4 Dashboard
- 3.2.5 Reservation
- 3.2.6 Reservation Confirm
- 3.2.7 Reservation Cancel

3.3 Modify front-end appearance

4. Deploy website
5. Create end-user guide
6. Create programming documentation for future students

4.6 Schedule

Our team has decided to use the AGILE style during the Spring semester to organize our tasks and stay on track with our goals. Our remaining sprints are outlined as follows:

Tasks	Dates
What Has Been Done • Discussed project complexity • Implemented models • Added unit tests • Added admin action functionality • Added external login functionality • Added reservation functionality • Mapped database with action URLs • Revamped user interface with Bootstrap	01/17 - 03/13
Sprint 5 (Final Code Sprint) • Final permission checking • End-user guide • Programming documentation in GitLab • Begin report/presentation	03/27 - 04/04
Sprint 6 (New Function Freeze) • Setup continuous deployment • Work with sponsors to test user cases • Make modifications/changes to app based on performance	04/05 - 04/18
Sprint 7 (Code Freeze)	04/19 - 05/02

• Technical contribution presentation • Final project modifications • Finalize end-user guide • Finalize programming documentation in GitLab • Finalize deployment • Final report/presentation	

4.7 Deliverables

- Design Document: This document will contain information about the models, views, methods, and functions that we will be implementing.
- Database schema: The DB schema is for an SQLite database and will be formally outlined in a comparable manner to the model design section in this proposal. An entity-relationship diagram will be created along with formal documentation of model/table interactions.
- End-User Guide: A help page will be created on the final website to walk users through the process of performing operations such as reserving a room or cancelling a reservation.
- Programming Documentation: We will be outlining various aspects of our design on the GitLab repository using GitLab wiki so that future students can better contribute to the project.
- Website code: The website will be coded as outlined in the design document over the course of the Spring semester and will be hosted in GitLab.
- Final Report
- Final Presentation Slides

5.0 Key Personnel

Creighton France – France is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed the Database Management Systems, Big Data Analytics and Management, Honors Programming Paradigms, and Software Engineering courses at the University. He is currently a Software Development Intern for GivePulse, Inc. and has previously worked with Python in research environments as well as other work environments. France was responsible for most of the communication between the team and the project sponsors, keeping the team on track with the development roadmap, and coordinating with IT for project deployment.

Alexandria Lim – Lim is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed software engineering. She has completed competitive internships at companies such as an information security internship at Pendo in Raleigh, North Carolina, and an application security internship at Regnology in Frankfurt, Germany. She worked on the infrastructure setup in coordination with IT services for email and OIDC login functionality and contributed to the report and documentation of the project.

Miguel Campbell – Campbell is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. In relevance to the project, he has completed courses including Software Engineering, Database Management Systems, and Programming Paradigms. He has prior experience with developing web applications, having worked alongside the Fall/Spring 20-21 Capstone team on the ReZerve project. Since June 2022, Campbell has been working at J.B. Hunt as an Applications Development Intern. Campbell was responsible for implementing the room reservation and email process, as well as contributing to the dashboard calendar.

Jackson DiRienzo – DiRienzo is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed courses such as software engineering, paradigms, and other pre-requisite courses. He has interned with Walmart's Cyber Security team and currently works for the University of Arkansas Housing Department. DiRienzo was responsible for implementing the dashboards visual and interactive elements that receive user engagement.

Wesley Parker – Parker is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed Database Management, Programming Paradigms, Algorithms, and Software Engineering. He has been interning with JB Hunt as an Application Development Intern since the summer of 2022. Parker was responsible for the report functions and the contextual actions in the admin dashboard.

Dr. Karl Schubert – Dr. Schubert is the Professor of Practice and Associate Director of the Data Science Program at the University of Arkansas and teaches in the program including the Data Science Practicum. After graduating with a PhD in Engineering from the University of Arkansas, he spent 35 years in industry before returning to the University to develop innovation and data science programs.

Lee Shoultz – Lee Schoultz is the program manager for the Data Science Program at the University of Arkansas. As program manager, she manages the day-to-day operations of the program as well as teaching DASC 1011 Success in Data Science Studies. Ms. Shoultz holds both a bachelor's and master's degree from the University of Arkansas. As an alum, Ms. Shoultz is passionate about the University of Arkansas and the symbol it represents in our state.

6.0 Facilities and Equipment

The Python web application will be created using the team's personal laptops, but the final web application should be hosted by IT Services, because it is being used by a formal University of Arkansas department. We will be deploying the webapp in a virtual machine on the University's servers, and the project code will be hosted in the University of Arkansas's chosen version control system GitLab.

7.0 References

[1] Herman, Michael. "Django vs. Flask in 2022: Which Framework to Choose." *Testdriven.io*, 24 Feb. 2022, https://testdriven.io/blog/django-vs-flask/.

[2] "Django Overview." *Django*, https://www.djangoproject.com/start/overview/.

[3] "Django Database Connectivity" *Tech Vidvan*

https://techvidvan.com/tutorials/django-database-connectivity/
[4] Teravainen, Taina. “Single Sign-On." *Tech Target,* https://www.techtarget.com/searchsecurity/definition/single-sign-on
[5] “Rooms in Mullins Library” *University of Arkansas Libraries*, https://libraries.uark.edu/rooms/
[6] “Reserve a Room” *University of Arkansas*, https://fulbright.uark.edu/departments/theatre/callboard/reserveroom.php
[7] “rezerve-startup/rezerve” *GitHub*, https://github.com/rezerve-startup/rezerve
[8] “Python Frameworks vs. Python Libraries.” *Full Scale*, 27 June 2022, https://fullscale.io/blog/python-frameworks-vs-python-libraries/
[9] Jain, Kalpit. “When to Use Django (and When Not to).” *Crowdbotics*, https://www.crowdbotics.com/blog/when-to-use-django-and-when-not-to.
[10] “Differences between Django vs Flask.” *GeeksforGeeks*, 15 June 2022, https://www.geeksforgeeks.org/differences-between-django-vs-flask/.