



Reserve-It

Online classroom booking application

Department of Information and Communication Technology
Faculty of Technology
University of Sri Jayewardenepura
Sri Lanka

Project Title

Reserve-It: An Online Classroom Booking Web-Application

Project Personnel

Academic Supervisor

Mr. Malshan Perera,
Lecturer,
Department of Information and Communication Technology,
Faculty of Technology,
University of Sri Jayewardenepura.
dinushamalshan064@gmail.com
+94 78 728 0357

Team Members

1. Dasanayaka L.K.K.S.
Multimedia Technology
Faculty of Technology
University of Sri Jayewardenepura
ICT/21/953
lct21953@fot.sjp.ac.lk
+94 710673062

2. Ekanayaka L.P.E.S.T.
Software Technology
Faculty of Technology
University of Sri Jayewardenepura
ICT/21/837
ict21837@fot.sjp.ac.lk
+94 761136338

3. Walimuni W.D.H.D.
Software Technology
Faculty of Technology
University of Sri Jayewardenepura
ICT/21/938
ict21938@fot.sjp.ac.lk
+94 774406113

4. Thilakarathna G.S.D.P.
Software Technology
Faculty of Technology
University of Sri Jayewardenepura
ICT/21/931
ict121931@fot.sjp.ac.lk
+94 765441217

5. Anuththara S.A.
Network Technology
Faculty of Technology
University of Sri Jayewardenepura
ICT/21/807
lct21807@fot.sjp.ac.lk
+94 716668336

1 Table of Contents

1	Table of Contents	3
1	Introduction and Background	4
2	System Requirements	5
2.1	Functional Requirements	5
2.2	Non-Functional Requirements	7
3	System Design	9
3.1	Main Functionalities	9
3.2	UI Wireframe	12
3.3	ER Diagram	18
3.4	Class Diagram	19
4	System Development, Testing and Deployment Strategies	20
4.1	System Development Strategy	20
4.2	Testing Strategy	22
4.3	Deployment Strategy	23
4.3.1	Development Deployment	23
4.3.2	Production Deployment	24
5	Project Milestones and Timeline	26

1 Introduction and Background

The Classroom Booking System is a web-based application designed to modernize and simplify the reservation process for classrooms and auditoriums at our university. This platform will offer an intuitive interface for students, faculty, and external users, allowing them to check room availability and make bookings in real-time. Key features include QR code access on classroom doors, an administrative review process for external bookings, and secure online payment options, all integrated with existing university systems for seamless operation.

This project aims to address the inefficiencies of our current system, providing a more user-friendly and secure experience. With role-based user authentication and robust payment processing, the Classroom Booking System will ensure data security and transaction safety. Our agile development team of five will deliver this project in a timely and organized manner, continually incorporating feedback to enhance the system. This new platform is set to significantly improve the booking experience, contributing to a more efficient and accessible university environment.

2 System Requirements

System requirements are divided into functional and non-functional requirements. Final Product must meet below mentioned functional and non-functional requirements in order to full fill the user needs. The basic system behavior is defined by the functional requirements.

2.1 Functional Requirements

- User Authentication and Authorization:
 - Role-based user authentication (students, faculty, administrators, and external users).
 - Secure login and registration system.
 - Password recovery and management.

- Room Availability Check:
 - Real-time display of available classrooms and auditoriums.
 - Search functionality based on date, time, and room capacity.
 - Calendar view to browse room availability.

- Booking Management:
 - Option to book classrooms and auditoriums for specific time slots.
 - Booking confirmation and notification system.
 - Option to modify or cancel bookings.
 - Administrative review process for external bookings.

- QR Code Access:
 - Generation of QR codes for booked classrooms.
 - Scanning functionality at classroom doors for access.

- Payment Processing:
 - Secure online payment gateway integration.
 - Support for multiple payment methods (credit card, debit card, digital wallets).
 - Transaction history and receipt generation.
- Integration with University Systems:
 - Sync with existing university scheduling and booking systems.
 - Integration with user directories (e.g., LDAP, SSO).
- User Interface:
 - Intuitive and responsive design using Next.js.
 - Mobile-friendly interface.
 - Dashboard for users to manage their bookings.
- Notifications and Alerts:
 - Email and SMS notifications for booking confirmations, cancellations, and reminders.
 - Alerts for upcoming bookings and changes in schedule.

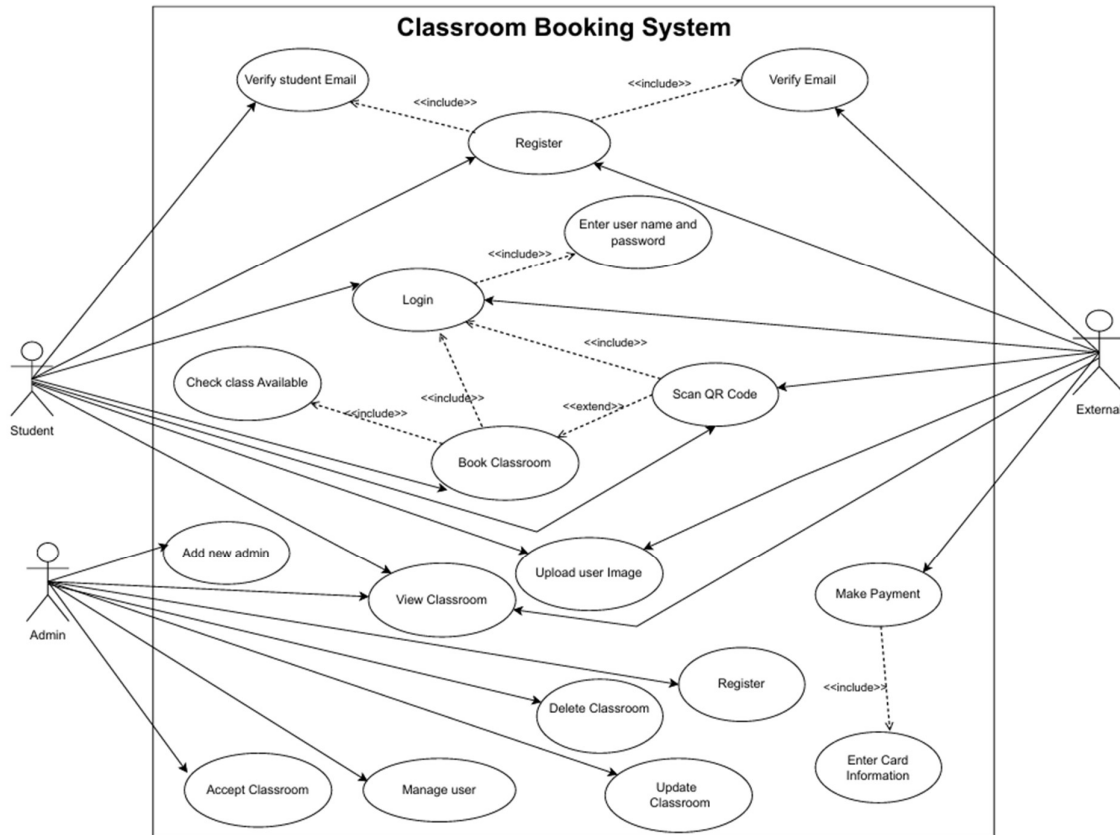


Figure 1: Use Case Diagram

2.2 Non-Functional Requirements

- Performance:
 - The system should handle up to 10,000 concurrent users without performance degradation.
 - Page load times should not exceed 3 seconds under normal load conditions.
 - System response time for booking operations should be less than 2 seconds.
- Scalability:
 - The system should be able to scale horizontally to accommodate increased load.
 - It should support automatic scaling based on demand.

- Availability:
 - The system should have an uptime of 99.9% to ensure availability for users.
 - Redundancy and failover mechanisms should be in place to maintain availability in case of server failures.
- Reliability:
 - The system should ensure data consistency and integrity during transactions.
 - It should handle failures gracefully, ensuring no data loss and minimal downtime.
- Security:
 - All data should be encrypted in transit and at rest.
 - The system should comply with university data protection and privacy policies.
 - Regular security audits and penetration testing should be conducted to identify and fix vulnerabilities.
- Usability:
 - The interface should be intuitive and user-friendly, requiring minimal training for users.
 - It should be accessible to users with disabilities, following WCAG 2.1 guidelines.
- Maintainability:
 - The system should be modular and well-documented to facilitate easy maintenance and updates.
 - Code should follow best practices and be easily understandable by new developers.

3 System Design

The process of designing the elements of a system, such as architecture, modules, and components, as well as how they interact with each other and the data that flows through the system, is known as the system design. This section will describe the basic components of the system and their interactions. Methods like Wire framing, ER Diagrams, Class Diagrams, and Activity Diagrams will be used.

3.1 Main Functionalities

This section will describe the design of the main functionalities of the "Reserve It" system.

1. Access to the Application

To access the application, users must log in to the system. If the user is registered, they can use their university email address, which was provided at the time of registration, to log in to the system. If the user is a new user, they must create a new account in the system by providing their university email address, name, and password. Once the user enters their email address, the system will send a verification link to that address and prompt the user to click the link to verify their email. Once the email is confirmed, the user will be redirected to the dashboard.

2. Checking Room Availability

The system allows users to check the availability of classrooms and auditoriums. Users can search for available rooms by specifying the date, time, and capacity requirements. The system will display a list of available rooms that meet the user's criteria, along with a calendar view for browsing availability. Users can click on a room to view detailed information, including location, capacity, and amenities.

3. Booking a Room

To book a classroom or auditorium, users must select an available time slot. Once a room is selected, the user will be redirected to a booking form where they can provide additional details such as the purpose of the booking and any special requirements. After filling out the form, the user can submit the booking request. The system will process the request and send a confirmation email with the booking details. Users can also modify or cancel their bookings from the dashboard.

4. QR Code Access

For each confirmed booking, the system will generate a unique QR code. Users can access this QR code from their dashboard or email. At the scheduled time, users can scan the QR code at the classroom door to gain access. The system will verify the QR code and unlock the door if the code is valid.

5. Payment Processing

For external users or bookings requiring payment, the system integrates a secure online payment gateway. Users can choose from multiple payment methods, such as credit cards, debit cards, or digital wallets. The system will process the payment and send a receipt via email. Users can view their transaction history and download receipts from their dashboard.

6. Integration with University Systems

The "Reserve It" application integrates seamlessly with existing university systems. It syncs with the university's scheduling system to avoid double bookings and utilizes the university's single sign-on (SSO) for user authentication. The system also integrates with the university's directory services (e.g., LDAP) to retrieve user information.

7. Notifications and Alerts

The system provides various notifications and alerts to keep users informed. Users receive email and SMS notifications for booking confirmations, cancellations, and reminders. The system also sends alerts for upcoming bookings and any changes in room availability.

8. User Interface

The "Reserve It" application features an intuitive and responsive user interface designed with Next.js. The interface is mobile-friendly, allowing users to manage their bookings on the go. The dashboard provides a centralized location for users to view and manage their bookings, access QR codes, and update their profile information.

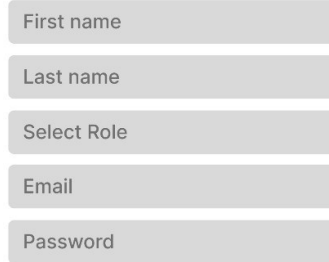
3.2 UI Wireframe

In this section wireframes are used to illustrate the system's structural design. The sizes and placements of page elements and application functionality will be described.



Figure 2: UI Wireframes and Flow

Register



First name

Last name

Select Role

Email

Password

Sign In

Figure 3: Register Page

Log In



Email

Password

Log in

Figure 4: Login Page

Welcome Admin

Add Lecture Room

Pending Request

Booking Request

Manage Users

Manage Lecture Rooms

Admin Information

Log Out

Add New Lecture Room

Lecture Hall No:

Lecture Hall Name:


Seats :

AC/ Non-AC:

ADD

Figure 5: Admin Dashboard

Lecture Hall Name



Print

Share

[Go to Dashboard](#)

Figure 6: Generating QR code

Welcome User

Book lecture room

Profile information

Booking history

First name:

last name:

Email :

Change Password :

New Password :

Conform Password :

Upload Image :

Update

Log Out

Figure 7: User Dashboard

Lecture Rooms

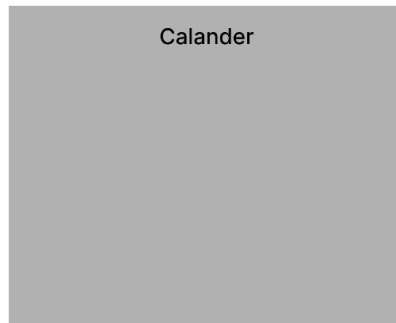
01	Lecture Hall name	100 Seats	See More
02	Lecture Hall name	100 Seats	See More
03	Lecture Hall name	100 Seats	See More
04	Lecture Hall name	100 Seats	See More
05	Lecture Hall name	100 Seats	See More
06	Lecture Hall name	100 Seats	See More
07	Lecture Hall name	100 Seats	See More

Figure 8: Classroom List Page



Lecture Room Details

Lecture Hall Name



Calander

Seats : 100

AC/Non-AC : AC

Book Now

Figure 9: Classroom Details Page



Book Lecture Room

Lecture Room name

Schedule

Start :

25/12/2024

8.00 AM

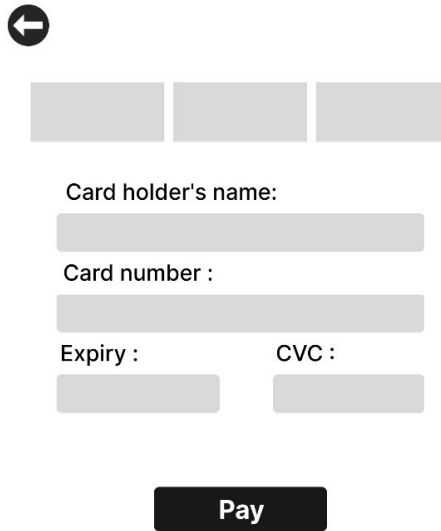
End :

25/12/2024

10.00 AM



Figure 10: Schedule Page



A payment form interface. At the top left is a circular back button with a white left-pointing arrow. Below it are three gray rectangular boxes for card details. Further down is the label 'Card holder's name:' followed by a gray input field. Below that is the label 'Card number :' followed by a gray input field. Then, 'Expiry :' and 'CVC :' labels are shown side-by-side, each followed by a gray input field. At the bottom center is a black button with the white text 'Pay'.

Figure 11: Payment Page

Your Booking Is Pending

Go To Dashboard

Figure 12: Status Page

3.3 ER Diagram

The ER diagram below illustrates the entities within the system and their interrelationships. This ERD will serve as a blueprint for database creation. ER diagrams are crucial in the development of high-quality databases.

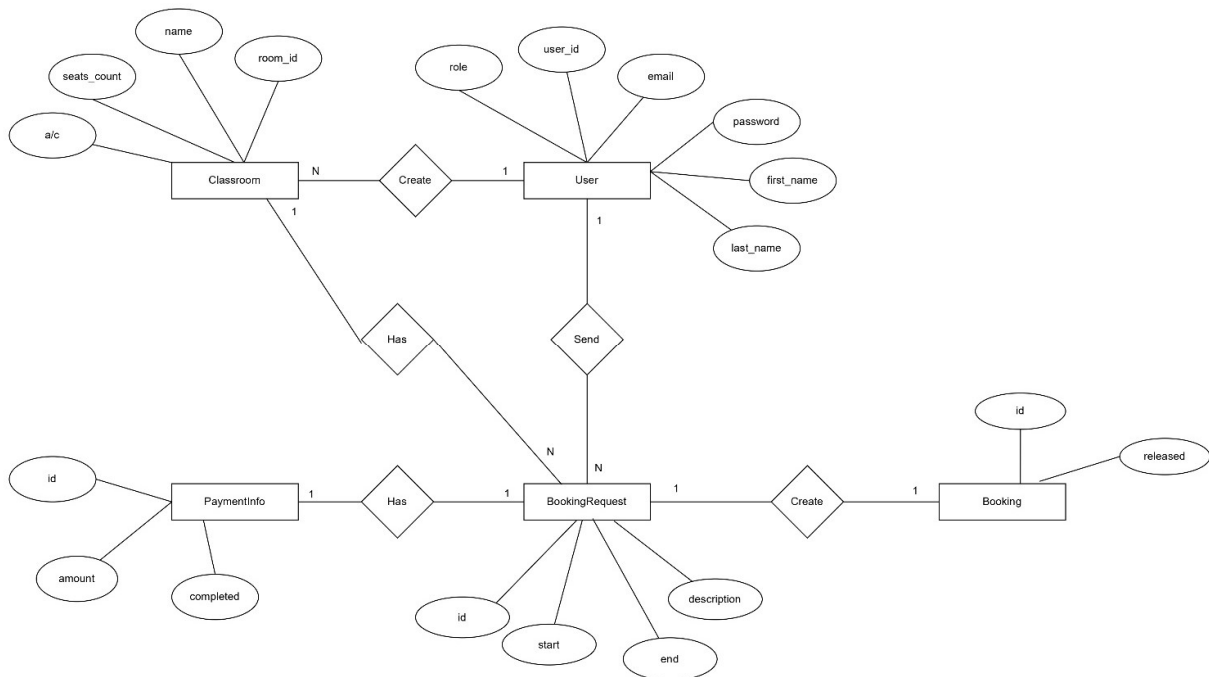


Figure 13: Entity Relation Diagram

3.4 Class Diagram

The class diagram below details the objects within the system, along with their attributes, behaviors, and the relationships between them.

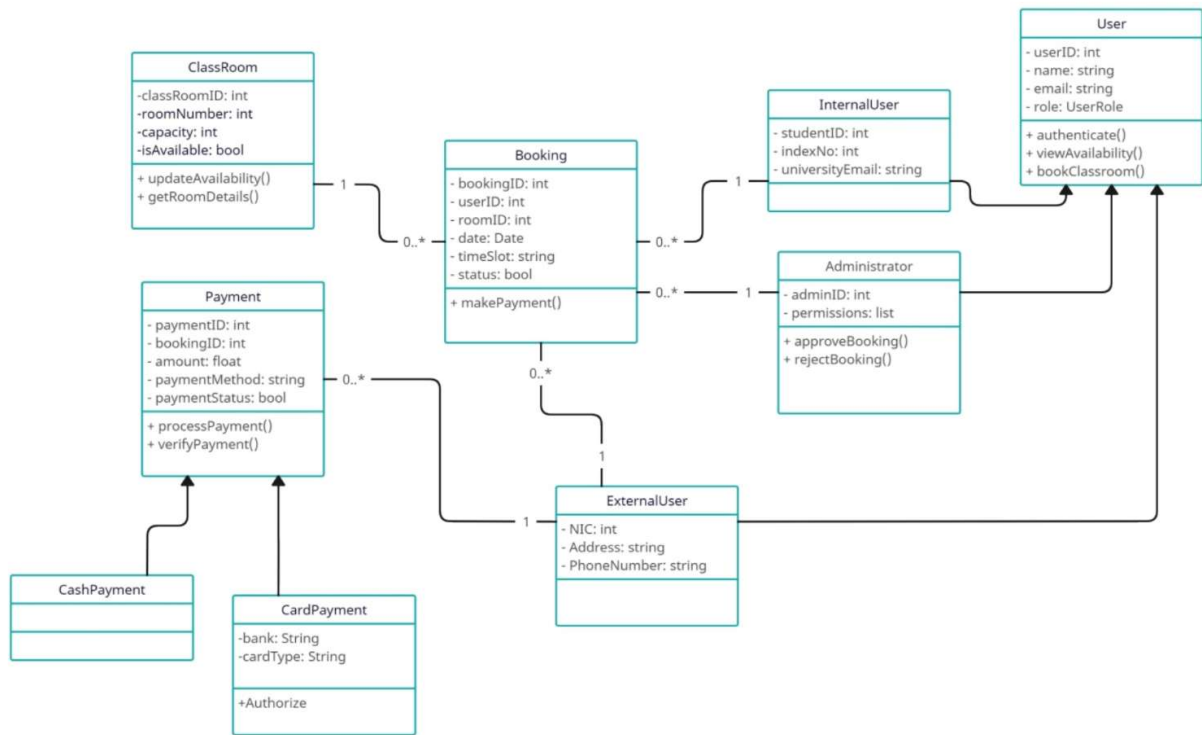


Figure 14: Class Diagram

4 System Development, Testing and Deployment Strategies

For your "Reserve It" classroom booking project, the system development, testing, and deployment strategies can be outlined as follows.

4.1 System Development Strategy

- Agile Development:
 - Sprint Planning: Organize work into one-week sprints with defined goals and deliverables.
 - Scrum Meetings: Daily stand-up meetings to discuss progress, roadblocks, and next steps.
 - Backlog Grooming: Regularly update and prioritize the product backlog based on feedback and changing requirements.
 - Iterative Development: Build the application incrementally, ensuring each increment adds value and is functional.
- Requirement Analysis and Documentation:
 - Functional Requirements: Document detailed user stories and use cases.
 - Non-Functional Requirements: Specify performance, security, and usability standards.
- Design and Architecture:
 - System Architecture: Define the overall system architecture, including frontend, backend, and integration points.
 - Database Design: Create an ER diagram and database schema based on system requirements.
 - User Interface Design: Develop wireframes and prototypes for user interfaces.

- Technology Stack:
 - Frontend Technologies
 - Next.js
 - Tailwind CSS
 - ShadCN
 - Backend Runtime
 - Node.js
 - Infrastructure
 - AWS
 - EC2
 - RDS (PostgreSQL)
 - VPC
 - NAT Gateway
 - Application Load Balancer
 - Authentication and Payments
 - Clerk
 - Stripe
 - Version Control and CI/CD
 - Git
 - GitHub
 - GitHub Actions
 - Docker
 - Development Tools
 - Visual Studio Code
 - Discord
 - Project Management
 - Jira

4.2 Testing Strategy

- Unit Testing:
 - Write and execute unit tests for individual components and functions using a testing framework like Jest.
- Integration Testing:
 - Test the interactions between different components and modules to ensure they work together as expected.
- System Testing:
 - Conduct end-to-end testing to validate the complete functionality of the application in an environment that simulates production.
- User Acceptance Testing (UAT):
 - Involve real users (students, faculty, and administrators) in testing the system to ensure it meets their needs and expectations.
- Performance Testing:
 - Perform load testing to ensure the system can handle expected user volumes and stress testing to find the system's breaking points.
- Security Testing:
 - Conduct security assessments, including vulnerability scanning and penetration testing, to identify and fix security issues.

4.3 Deployment Strategy

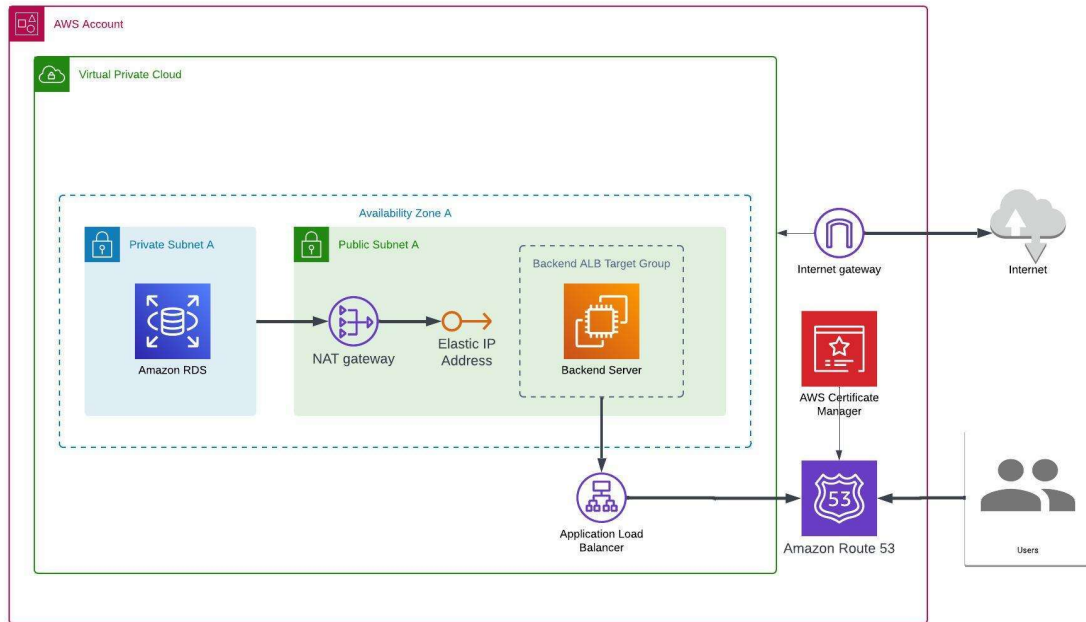


Figure 15: Datacenter Network

4.3.1 Development Deployment

For development purposes, we utilize Vercel to streamline the deployment process, allowing for quick iterations and feedback. The application is connected to a public RDS instance on AWS, ensuring that our development team can work with a live database environment similar to production.

Key Components:

- **Vercel:** Hosts the Next.js application, providing seamless deployment and easy rollbacks.
- **Public RDS Instance (AWS):** Allows for real-time database interactions during development.

4.3.2 Production Deployment

Our production deployment strategy focuses on leveraging AWS infrastructure to ensure security, scalability, and high availability. The deployment involves setting up both public and private subnets within a Virtual Private Cloud (VPC), with specific roles assigned to each component of the application.

Key Components:

- Private Subnet:
 - RDS Instance (PostgreSQL): Hosted in a private subnet to enhance security. The database is not directly accessible from the internet.
 - NAT Gateway: Allows resources in the private subnet to access the internet for updates and patches without exposing them directly.
- Public Subnet:
 - EC2 Instances: Host the web application. Multiple instances ensure load distribution and high availability.
 - Application Load Balancer: Distributes incoming traffic across multiple EC2 instances, ensuring reliability and performance.
- Staging Environment:
 - Mirrors the production setup but utilizes only one EC2 instance to
 - reduce costs while still providing a realistic testing environment.

Workflow:

1. VPC Configuration:
 - a. Set up a VPC with both public and private subnets.
 - b. Configure security groups and IAM roles to manage access.
2. Database Deployment:
 - a. Deploy the RDS instance in the private subnet.
 - b. Secure access to the database through security group settings.

3. Application Deployment:

- a. Launch EC2 instances in the public subnet.
- b. Deploy the Next.js application on the EC2 instances.
- c. Configure the Application Load Balancer to distribute traffic.

4. Staging Deployment:

- a. Deploy a single EC2 instance with a similar configuration to production for testing purposes.

5. Continuous Integration/Continuous Deployment (CI/CD):

- a. Use GitHub Actions to automate the build, test, and deployment processes.
- b. Ensure smooth and error-free deployments to both staging and production environments.

User Training and Support:

- Provide training sessions and documentation for end users.
- Set up a support system to handle user queries and issues post-deployment.

5 Project Milestones and Timeline

The project aims to complete within a timeframe of 13 weeks. Adopting agile development methodology means that testing and validation will occur incrementally alongside development. The development of the application itself is expected to be completed within a maximum of 7 weeks, with the remaining activities scheduled over the subsequent 8 weeks.

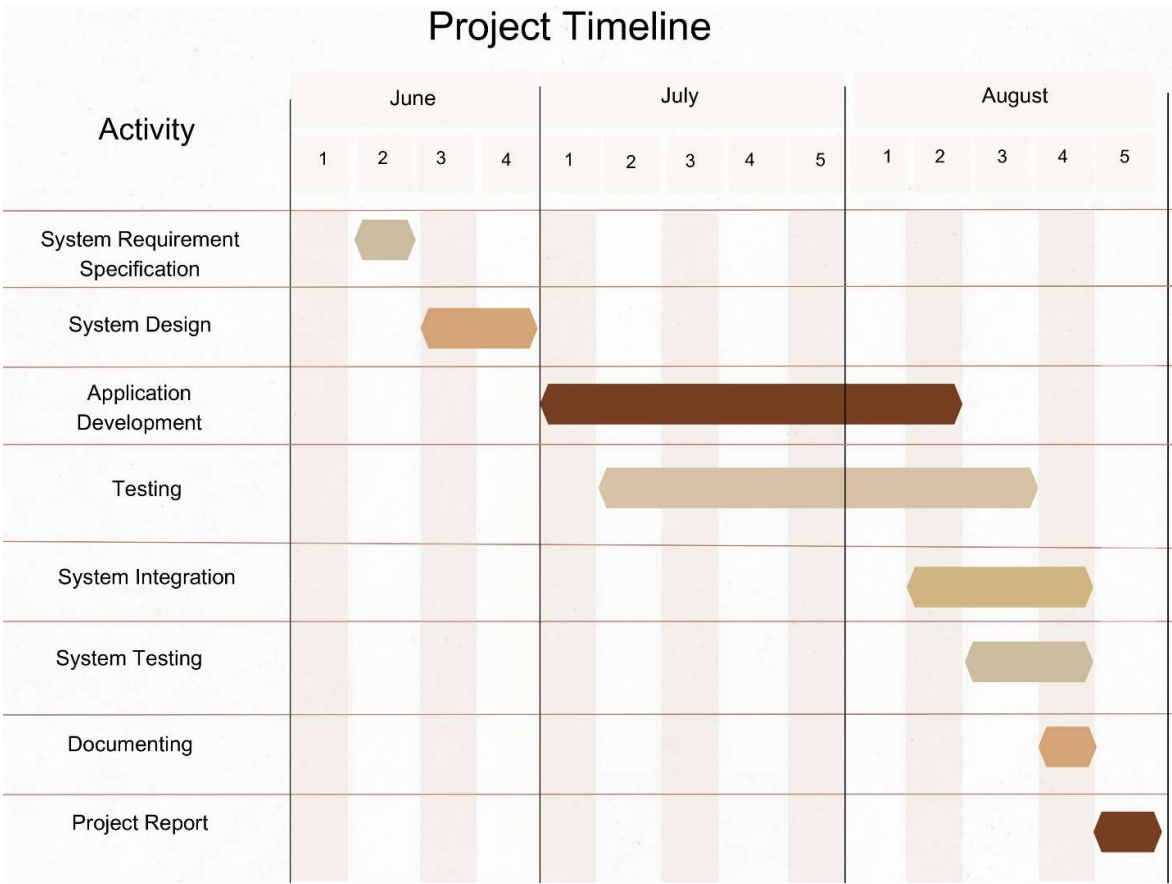


Figure 16: Project milestones and Timeline