Software Requirements Specification for BeAvis Car Rental System

Prepared by: Arman Atwal, Sahib Singh, Catherine Dang

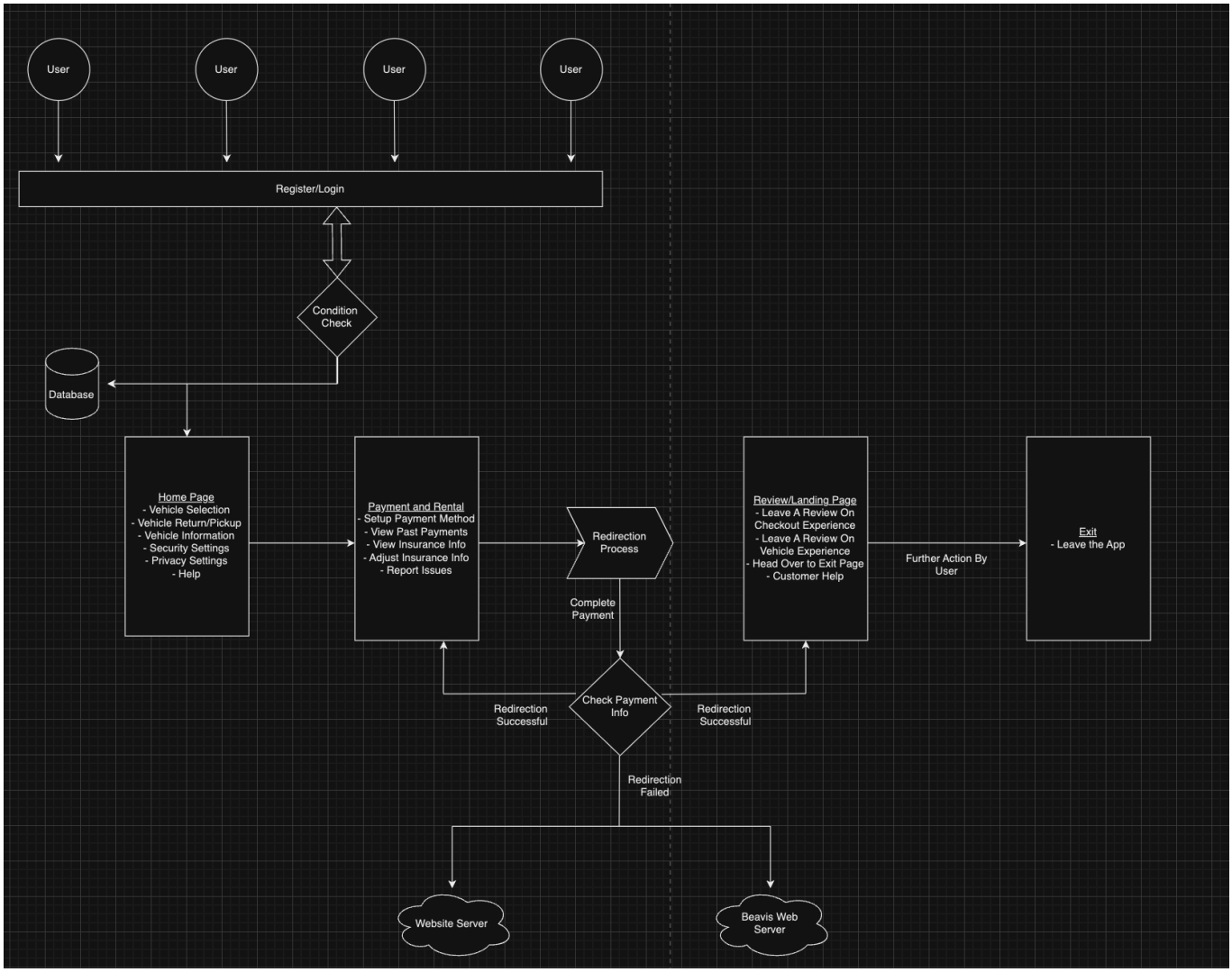
System Description

The BeAvis Car Rental System represents a cutting-edge solution meticulously crafted to revolutionize the car rental experience. It caters to two distinct user categories: customers, seeking a hassle-free vehicle rental process, and employees, entrusted with the efficient management of rental operations. Accessibility is at the core of our design ethos, ensuring that users can seamlessly engage with the system via mobile applications compatible with both iOS and Android, in addition to web browsers across diverse computer platforms. Our unwavering commitment to user-friendliness is reflected in a feature-rich environment encompassing user account creation, an intuitive vehicle selection process, comprehensive vehicle information presentation, and a robust and secure payment processing framework.

Legal and regulatory compliance is paramount, particularly with regard to data privacy laws. To fortify security, the system offers optional two-factor authentication, a robust safeguard for user accounts. Furthermore, it supports a wide array of payment methods, ensuring the strict segregation of user data and sensitive payment information. Real-time responsiveness is integral, with the system engineered to handle user requests promptly and furnish instantaneous updates on vehicle availability. Scalability has been meticulously considered, with the capacity to accommodate up to 15,000 concurrent users and 5,000 concurrent rentals, ensuring seamless expansion in tandem with business growth.

The technology stack underpinning the BeAvis Car Rental System has been thoughtfully selected to maximize development productivity and operational efficiency. In essence, the system embodies a commitment to efficiency, stringent security, and sustainable revenue growth, poised to establish itself as an industry leader in the fiercely competitive automobile rental sector.

Software Architecture Overview



Description of SWA for Users:

The Software Architecture Diagram for the BeAvis system provides a clear and concise representation of how users interact with the system and the various components involved in the process. It effectively outlines the user journey from registration or login to completing a rental and leaving reviews. Here's a brief breakdown of the components and flow in our diagram:

User Registration/Login:

- Users start their journey by either registering for a new account or logging into an existing one.

Database:

- Successful login or registration grants users access to the database, where their account information is stored.

Home Page:

- Once authenticated, users are directed to the home page, which serves as the central hub for various actions and services.
- Home Page Components:
    - Vehicle Selection
    - Vehicle Return/Pickup
    - Vehicle Information
    - Security Settings
    - Privacy Settings
    - Help

Payment Details/Rental:

- From the home page, users can initiate the rental process, including setting up payment methods, viewing past payments, accessing insurance information, adjusting insurance details, and reporting issues.

Redirection Process:

- After setting up payment details, users are redirected to complete their payment. At this stage, the system verifies the payment information provided by the user.
- If the payment information is incorrect, users are sent back to the payment page to make corrections.
- If the payment information is correct, users proceed to the review page.

Review Page:

- The review page allows users to provide feedback on their checkout experience, vehicle experience, and access customer help if needed.
- Review Page Components:
    - Leave A Review On Checkout Experience
    - Leave A Review On Vehicle Experience
    - Head Over to Exit Page
    - Customer Help

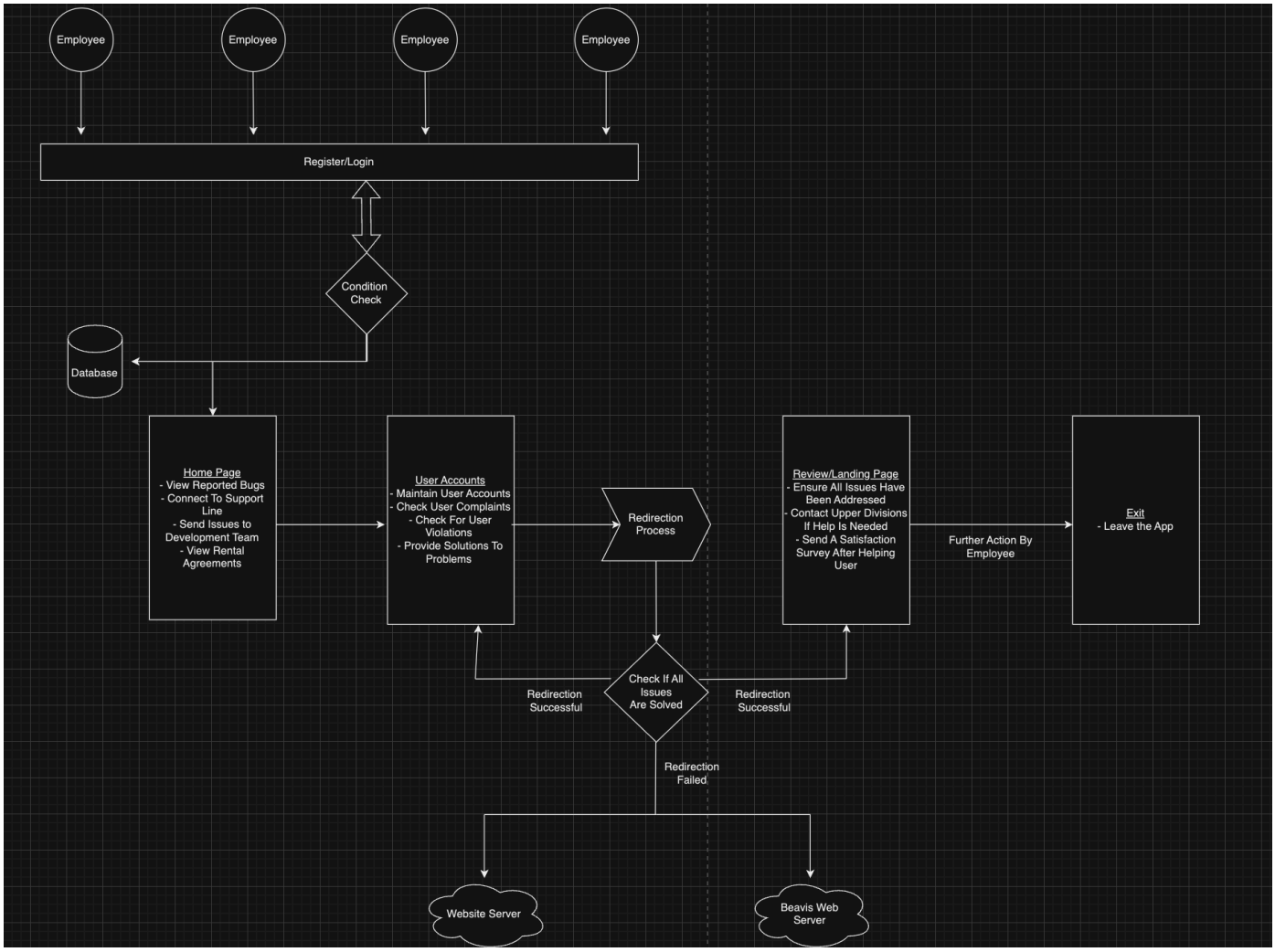
Exit Page:

- The exit page marks the conclusion of the user journey within the system.

Website Server and BeAvis Web Server:

- The entire process is connected to both the website server and the BeAvis web server, which handle the processing and communication required to facilitate user interactions.

Software Architecture Overview Continued



Description of SWA for Employees:

The Software Architecture Diagram for the BeAvis system provides a clear and organized visualization of how employees navigate and interact with the system. It effectively illustrates the flow of actions and decisions that employees can take within the system. Here's a breakdown of the components and flow we've described:

User Authentication Flow:

- Employees can either register for an account or log in to the system.
- Successful authentication grants access to the database and directs them to the home page.
- Unsuccessful login attempts prompt employees to retry the login process.

Home Page:

- The home page serves as the central dashboard for employees and provides quick access to key functionalities:
    - View Reported Bugs: Access to information about reported issues.
    - Connect To Support Line: A feature for immediate assistance.
    - Send Issues to Development Team: The ability to escalate issues to the development team.
    - View Rental Agreements: Access to rental agreements for reference.

User Accounts:

- Employees can navigate to the user accounts section, which encompasses various management tasks:
    - Maintain User Accounts: Employee access to user account management functions.
    - Check User Complaints: Monitoring and reviewing user complaints.
    - Check For User Violations: Reviewing any user violations or breaches of terms.
    - Provide Solutions To Problems: Resolving user issues and concerns.

Redirection Process:

- After interacting with user accounts, there is a redirection process based on the outcome:
    - If user issues are resolved, employees proceed to the review page.
    - If not, they are directed back to user accounts for further action.

Review Page:

- The review page focuses on ensuring the satisfactory resolution of user issues:
    - Ensure All Issues Have Been Addressed: Confirming that all user concerns are adequately resolved.
    - Contact Upper Divisions If Help Is Needed: The ability to escalate issues further.
    - Send A Satisfaction Survey After Helping User: Collecting feedback from users to gauge their satisfaction with the resolution.

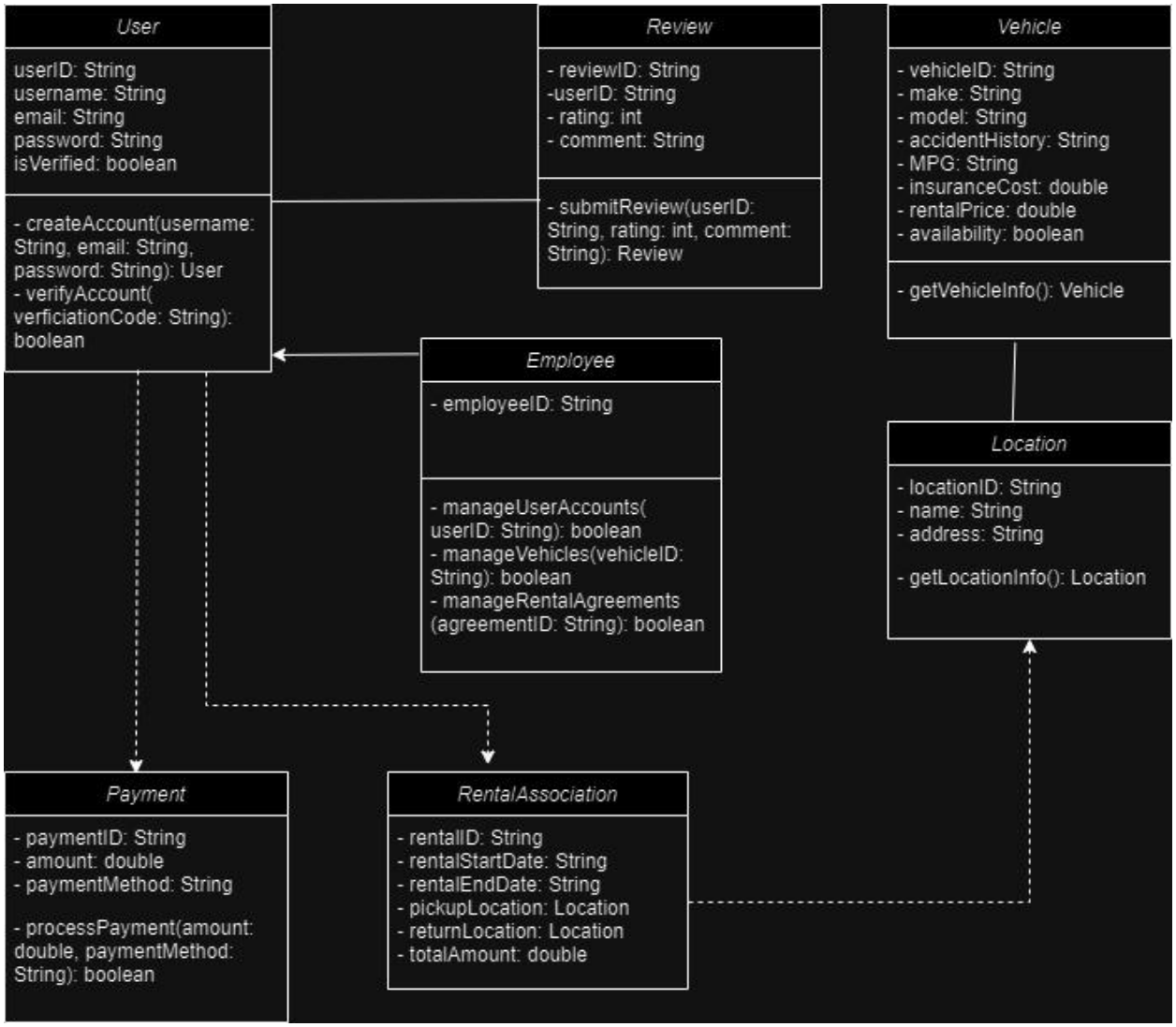
Exit Page:

- The exit page marks the conclusion of the employee's session within the system.

Server Connectivity:

- The entire process is connected to both the website server and the BeAvis web server, ensuring seamless data exchange and functionality

UML Class Diagram



Description of UML Class Diagram

The UML diagram for the BeAvis Car Rental System is designed to capture the system's structure and interactions, ensuring a streamlined user experience.

- The User Class serves as the foundation, which has the necessary attributes such as userID(String), username(String), email(String), password(String), and isVerified(boolean). Users engage with the system through defined function interfaces, such as createAccount(username: String, email: String, password: String): User, allowing the creation of accounts and verifyAccount(verificationCode: String): boolean for secure verification processes.
- The Vehicle Class contains the details of available rental cars, which includes properties like vehicleID(String), make(String), model(String), accidentHistory(String), MPG(String), insuranceCost(double), rentalPrice(double), and availability(boolean).

Users interact with vehicles via functions like getVehicleInfo(): Vehicle, enabling them to retrieve information before making rental decisions.

- Rental locations are represented by the Location Class, featuring attributes like locationID(String), name(String), and address(String). The getLocationInfo(): Location function interface manages access to location details, helping users in choosing convenient pickup and return points.
- For smooth transactions, the Payment Class manages financial components, storing paymentID(String), amount(double), and paymentMethod(String). The system ensures secure payments through the processPayment(amount: double, paymentMethod: String): boolean function interface, where users initiate transactions with specified amounts and payment methods.
- User feedback is facilitated by the Review Class, which has attributes such as reviewID(String), userID(String), rating(int), and comment(String). Users can submit reviews via the submitReview(userID: String, rating: int, comment: String): Review function, enhancing transparency and trust within the system.
- The RentalAssociation Class manages the rental process, which includes details like rentalID(String), rentalStartDate(String), rentalEndDate(String), pickupLocation(Location), returnLocation(Location), and totalAmount(double). This class ensures a comprehensive record of each transaction, facilitating efficient management by both users and employees.
- The Employee Class, a subclass of User, includes specialized attributes such as employeeID(String). Employees perform administrative tasks via functions inherited from the User class, enabling them to manage user accounts, vehicle inventory, and rental agreements effectively.

Development Plan, Responsibilities, and Timeline

Creating a comprehensive developer plan, responsibility, and timeline for building the BeAvis Car Rental System requires careful consideration of the project's complexity and the responsibilities of the development team. Here's a high-level plan with roles and a timeline for a programmer working on this system:

Project Duration: 12 months

Roles and Responsibilities:

Lead Developer/Architect (Programmer):

- Overall technical ownership of the project.
- Define the system's architecture, design patterns, and technology stack.
- Develop and oversee the implementation of critical system components.
- Ensure code quality, scalability, and security.
- Collaborate with other team members for integration and testing.

Frontend Developer (Programmer):

- Develop user interfaces for web and mobile applications (iOS and Android).
- Implement the user registration and login process.
- Create the home page and its components (vehicle selection, payment setup, etc.).
- Develop the review and exit pages.

Backend Developer (Programmer):

- Set up and maintain the server infrastructure.
- Implement the database schema and handle data storage.
- Develop APIs for user authentication and interaction with user accounts.
- Implement payment processing and integration with payment gateways.
- Create APIs for employee functionalities (user account management, issue resolution, etc.).

Database Administrator (DBA):

- Design and optimize the database schema.
- Ensure data integrity, security, and backup procedures.
- Optimize database performance for high concurrency.
- Collaborate with backend developers for data retrieval and storage.

Quality Assurance Engineer (QA):

- Create test plans, test cases, and test data.
- Perform manual and automated testing of the system.
- Identify and report bugs and issues.
- Ensure the system complies with legal and regulatory requirements.

DevOps Engineer:

- Set up continuous integration/continuous deployment (CI/CD) pipelines.
- Manage server deployment and scaling.
- Ensure system availability, monitoring, and logging.
- Collaborate with other developers to optimize performance.

Timeline:

Month 1-2: Project Initiation and Planning

- Define project scope, requirements, and objectives.
- Create a detailed project plan and timeline.
- Select the technology stack and tools.
- Set up the development environment and version control system.

Month 3-4: System Architecture and Backend Development

- Design the system architecture.
- Develop the backend infrastructure, including the server and database.
- Implement user authentication and registration.

Month 5-6: Frontend Development

- Create user interfaces for web, iOS, and Android.
- Implement the home page and its components.
- Develop the vehicle selection and payment setup features.

Month 7: Payment Processing and Integration

- Implement payment processing and integrate with payment gateways.
- Develop the redirection process for payment.
- Test payment flows and security.

Month 8-10: Employee Functionalities and Testing

- Develop employee-specific features for user account management.
- Implement issue resolution and escalation functionalities.
- Conduct thorough testing, including security and compliance checks.

Month 11: Final Testing and Quality Assurance

- Perform comprehensive testing, including load testing and security audits.
- Identify and address any remaining bugs and issues.
- Ensure compliance with data privacy laws.

Month 12: Deployment and Launch

- Set up production servers and databases.
- Deploy the system to production.
- Monitor system performance and address any post-launch issues.
- Launch the BeAvis Car Rental System to customers and employees.

This plan is a high-level overview and can be adjusted based on the project's specific requirements and resources. It's essential to have regular meetings, code reviews, and quality assurance throughout the development process to ensure a successful and efficient project delivery.