# CS 255 System Design Document Template

Tony Allam

#### **Purpose**

The purpose of the DriverPass system is to help students pass their driving test by giving them access to online practice exams and scheduling real driving lessons. This system will also allow instructors and administrators to manage lessons, students, and training content.

#### **System Background**

DriverPass discovered that more than 65% of people fail the driving test because they only study old test questions. They want a better way for students to learn and prepare. Right now DriverPass does not have a working system, so everything will be new. The system needs to support students signing up, taking practice tests, getting feedback, and booking driving lessons with qualified instructors.

#### **Objectives and Goals**

The main goals of the DriverPass system are:

• Help students prepare and pass the driving test

• Allow easy scheduling and management of driving lessons

• Provide secure and reliable practice tests online

• Notify students about lesson times, changes, and results

• Give administrators control over instructors, cars, and test content

• Offer a better experience than studying old questions

## UML Diagrams

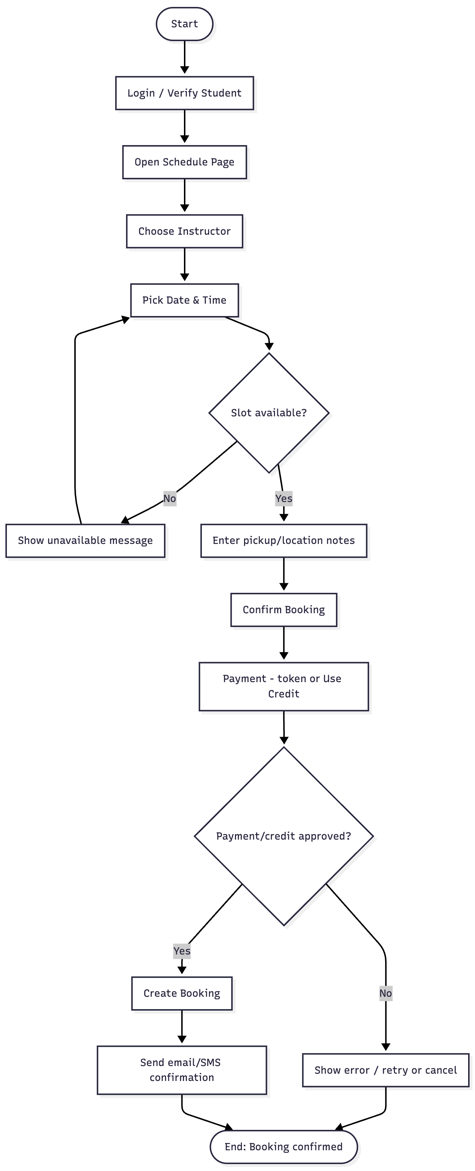
### UML Use Case Diagram

A diagram of a flowchart

Description automatically generated

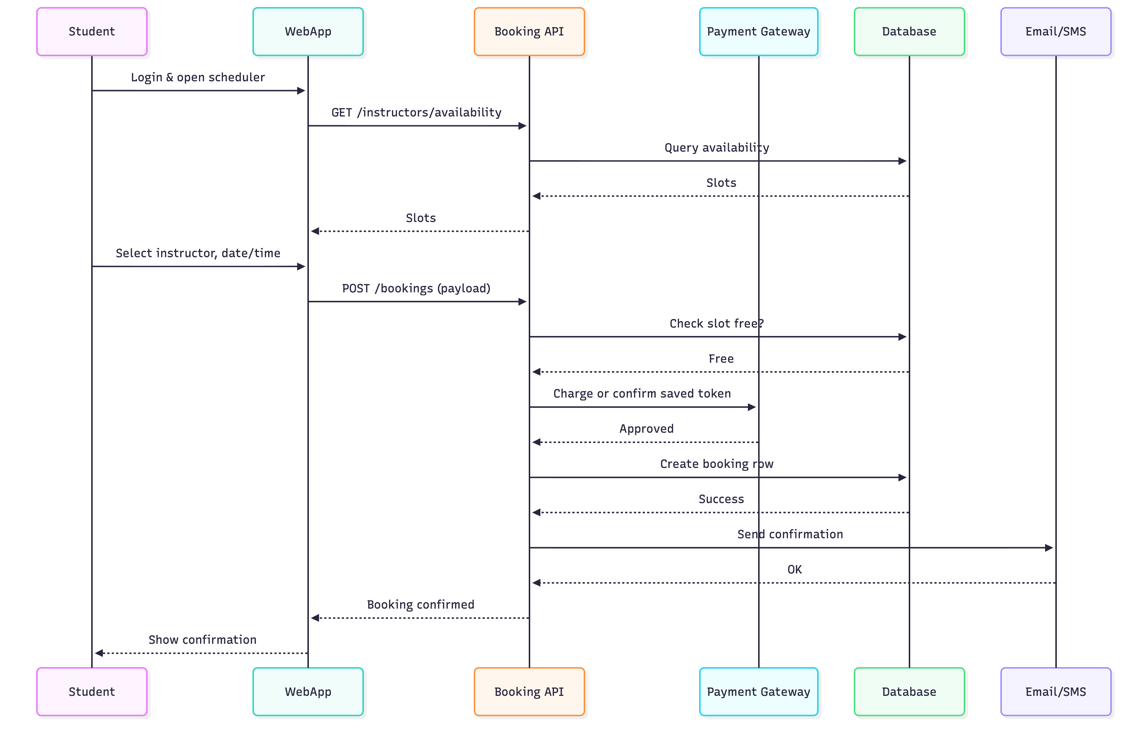
### UML Activity Diagrams

## Activity 1- Schedule Driving Lesson Activity Diagram 2 — Take Online Practice Test

A diagram of a diamond with text

Description automatically generated with medium confidence

## UML Sequence Diagram - Schedule Driving Lesson



### UML Class Diagram- Core Model

A diagram of a computer program

Description automatically generated

## Technical Requirements

DriverPass will be a secure, cloud-hosted, web-based system that allows students to take practice tests, schedule on-road lessons, make payments, and receive email/SMS notifications.

### System Architecture

* **Web Application:**
  + Responsive browser-based UI (HTML5, CSS3, JavaScript, React or similar).
  + Accessible design (WCAG 2.1 AA).
* **Backend API:**
  + RESTful services for authentication, scheduling, tests, results, and payments.
  + Implemented using a secure, server-side framework (Node.js/Express, Java/Spring Boot, or .NET).
* **Database:**
  + Relational database such as MySQL/PostgreSQL.
  + Stores users, instructors, questions, availability, bookings, and test attempts.

### Infrastructure

* Hosted on a cloud platform (AWS/Azure/GCP).
* Components: Load balancer, application servers, managed SQL database.
* Auto-scaling to support growth in student usage.

### Third-Party Integrations

* **Payment processing:** Stripe or similar service (tokenized card data, PCI-DSS compliant).
* **Notifications:** Email/SMS services such as SendGrid or Twilio.

### Security Requirements

* Secure login with password hashing (bcrypt or Argon2).
* Role-based access control (Student, Instructor, Admin).
* All communication encrypted with HTTPS/TLS.
* Input validation and protection against common attacks (SQL injection, XSS).
* Sensitive configuration stored securely (no hard-coded credentials).

### Performance & Availability

* Target uptime: 99.5%.
* System supports 500+ concurrent user**s** initially.
* Fast response time (API requests p95 ≤ 300ms under typical load).
* Daily automatic database backups with point-in-time recovery.

### Compatibility

* Supported browsers: Chrome, Safari, Firefox, Edge (current + previous version).
* Mobile-friendly interface for smartphones and tablets.

### Maintenance and Monitoring

* Logging and health monitoring for API performance and errors.
* Alerts for failed payments, notification failures, and service outages.
* Scheduled updates and patching through automated deployments (CI/CD).