**ONLINE CANDIDATE VERIFICATION SYSTEM**

**PROJECT ROADMAP**

1. **Project Setup:**

- Install Django: Make sure you have Django installed. You can use `pip` to install it: `pip install Django`.

- Create a New Django Project: Use `django-admin` to create a new project: `django-admin startproject ExamVerificationSystem`.

1. **Create App:**

- Inside the project folder, create a new app: `python manage.py startapp verification`.

1. **Define Models:**

- Define models to represent entities like `Candidate`, `ExamHall`, `MalpracticeReport`, etc. in the `models.py` file of the `verification` app.

1. **Admin Panel:**

- Register the models in the admin panel (`admin.py`) to manage data through the Django admin interface.

1. **Views and Templates:**

- Create views for candidate registration, exam hall verification, malpractice reporting, etc.

- Design HTML templates for the views to provide a user-friendly interface.

1. **URL Configuration:**

- Define URLs and map them to the corresponding views using `urls.py`.

1. **Authentication:**

- Implement secure candidate authentication mechanisms. You can use Django's built-in authentication system or third-party packages like `django-allauth`.

1. **Database Configuration:**

- Configure your database settings in `settings.py`. You can use PostgreSQL, MySQL, SQLite, etc.

1. **Business Logic:**

- Implement the business logic to perform candidate registration verification, real-time hall capacity monitoring, malpractice reporting, etc.

1. **Real-Time Updates:**

- To achieve real-time hall capacity monitoring, you might need to use technologies like WebSockets or AJAX to update occupancy information without requiring full page reloads.

1. **Mobile Application or Web Portal:**

- Depending on your expertise, you can choose to create either a mobile application using a framework like Django REST framework and a frontend framework like React Native, or you can create a responsive web portal that works well on mobile devices.

1. **Testing:**

- Write unit tests and integration tests to ensure the functionality is working as expected.

1. **Deployment:**

- Deploy the application to a hosting platform of your choice. Popular options include Heroku, AWS, and DigitalOcean.

1. **Security:**

- Implement security measures to protect user data, prevent unauthorized access, and guard against potential vulnerabilities.

1. **Documentation:**

- Document the codebase, APIs, and any setup instructions for other developers who might work on the project.

1. **User Testing and Feedback:**

- Test the application thoroughly and gather feedback from potential users to identify any usability issues or bugs.

Remember that this is a high-level overview, and each step involves a significant amount of work and decision-making. Be prepared to dive into specific Django concepts, HTML/CSS, and potentially frontend or mobile development depending on the platform you choose for the user interface. Also, consider keeping security and user privacy as top priorities throughout the development process.